Now Bimonthly!

SEXTANT

$2.95
March–April 1984
Issue No. 9

The Independent Magazine for the Entire Zenith Computer Community and Users of Zenith-Compatible Computers

Why Won't All That IBM Personal Computer Software Run on My Z100?

Screen Dump!
Build a Sound/Clock Board for Your '89
Drawing on Your Heath/Zenith Computer
Clarkson College Puts a Z100 on Every Desk
How One Family Learned to Live with an H89

SEXTANT

Features

Photo Courtesy of Zenith Data Systems

8

When marketing the Z100, Zenith Data Systems has emphasized its non-technical, "user-friendly" aspects. Yet it has scored its major coups among people who prize high-tech excellence above most other considerations. The most notable example of this phenomenon came when Clarkson College of Technology agreed to buy a thousand Z100s a year. You might have seen the story on ABC TV or in a national wire-service article. Here's the background and the forecast—what happened long before the TV cameras rolled up, and what will be going on long after it's no longer "tonight's trend."

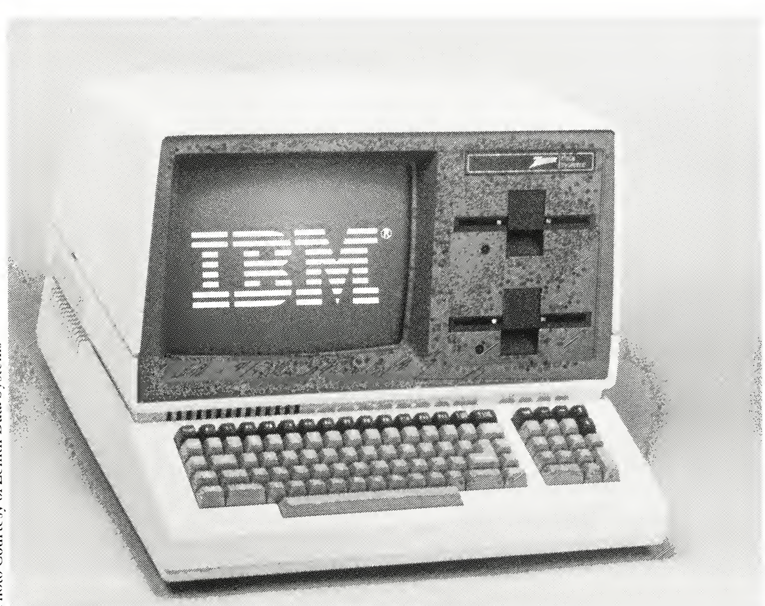


Photo Courtesy of Zenith Data Systems

51

All right, so a lot us have always had a thing against IBM. But the IBM Personal Computer has all that nice *software* written for it. And theoretically, it *ought* to run on the H/Z100. But (as many of us have found out) it usually *won't*. The author of this article explains why, what you'd need to get the programs to run, and why some problems will probably remain. He guides you through the thicket of programmer "shortcuts" he encountered while stalking the wild interrupt.

Issue No. 9
March-April 1984

Departments

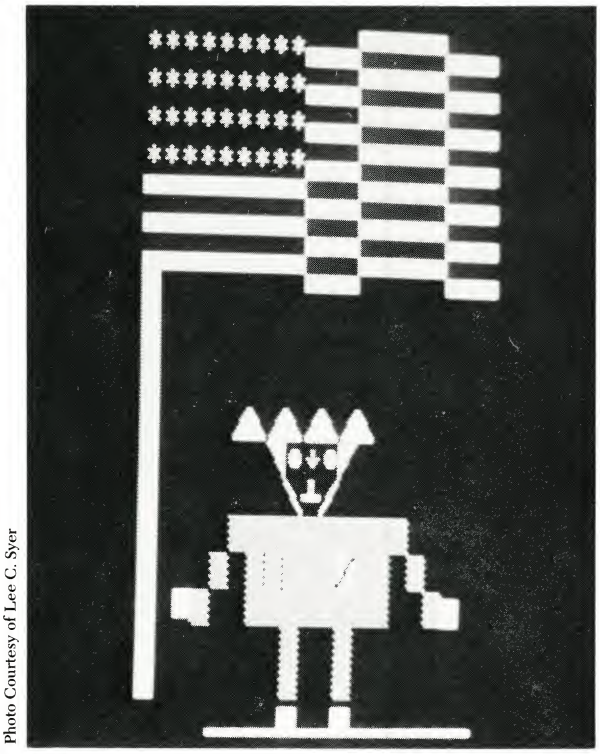


Photo Courtesy of Lee C. Syer

17 For some brands of computers, "graphics" means adding hardware so that the computer can display more than just text. But the H19 and H/Z89 have a graphics character set that can be used for a wide range of graphics needs without adding any hardware. Ed-A-Sketch is a popular program to make the Heath/Zenith's native graphics ability as convenient and flexible as it can be.

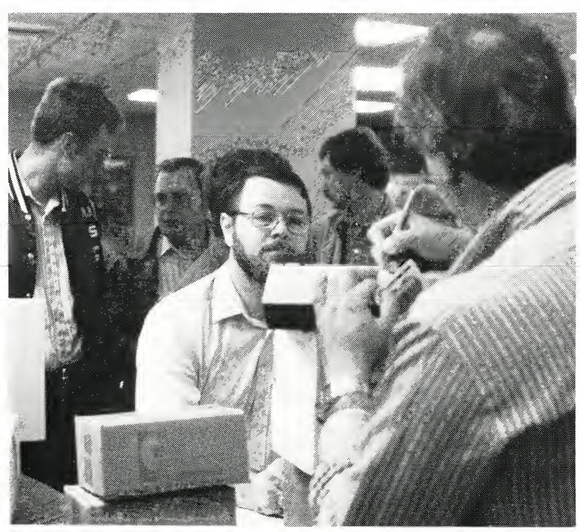


Photo by Charles Floto

41 HUGgies who went to CHUGCON 83 benefitted from seminars, vendor exhibits, classes, speeches, and each other. It was all very sociable and high tech. But in the midst of it, some of the heaviest applause went to Terry Jensen for pointing out that old-fashioned ethics and responsibility still have an important part to play, too.

Circle #100 on Reader Service Card

SEXTANT

Charles Floto
Editor
Publisher

Sharon E. Conaway
Assistant Publisher
Advertising Manager

Elizabeth P. Saxon
Production Manager
Marketing Assistant

John Walker
Technical Editor

Jennie D. Allen
Assistant Editor
Production Assistant

Frederick Zimmerman
Staff Writer

Martha W. Swain
Production Assistant
Circulation Assistant

John Gill
Circulation Assistant

Subscription Service • 202/544–0900

Advertising Inquiries • 202/544–2868

Authors' Inquiries • 202/544–2867

Charles Floto • 202/544–0484

Sextant (ISSN 0731–2180) is published bimonthly by Sextant Publishing Co., 716 E Street SE, Washington, DC 20003. Second-class postage paid at Washington, DC and at additional mailing offices. POSTMASTER: Send address changes to Sextant, 716 E Street SE, Washington, DC 20003.

 Printed in the USA by The Sheridan Press, Hanover, PA 17331.

Subscription rates: 6 issues (1 year) in U.S.—$14.97; in Canada and Mexico—$17.25; overseas—$21.00. Payment accepted by check in U.S. dollars payable on a U.S. bank, by U.S. and international postal money order, and by Visa and MasterCard. Please allow six to eight weeks for delivery.

The Editorial Eye

There's no such thing as a monthly magazine. The substitute is to publish three different quarterly magazines under the same title, and sell them under a joint subscription. Each issue takes about three months to produce, with additional time spent editing the articles.

Similarly, by the time the copies of this March-April *Sextant* start reaching subscribers, we'll be well into the production process for the May-June issue. We'll also be working on typesetting for the July-August *Sextant.* We're scheduled to start planning the contents of the September-October issue in late March. (Tentative plans for 1984's staff-written articles were drawn up last July.)

Articles we're preparing for future issues include:

- a survey of hardware to produce color graphics with an H8 or H/Z89
- enhancements to the HDOS device driver for Diablo printers
- notes on starting a local Heath Users' Group
- a BASIC program to turn your computer system into a correctable, previewing typewriter
- a roundup of educational software for CP/M, Z-DOS, and HDOS

We try to have a good balance of articles in each issue, as well as serving the wide range of interests within the Zenith computer community in the six issues we'll be publishing annually. As indicated on the cover of this issue, we also want to serve users of Zenith-compatible computers. Two companies advertising in this *Sextant* have announced plans to produce computers compatible with the H8 and H/Z89. I have reason to believe that, by the time you read this, there will also be a Zenith-compatible computer from IBM.

We'll bring you news of all these developments as soon as our production schedule permits. Some other magazines feel a bit more pressure to rush into print with the latest rumors or advance product announcements. At *Sextant,* we leave such coverage to our sister publication. It travels by first class mail 20 times a year under the name of *Buss: The Independent Newsletter of Heath Co. Computers.* Each issue typically is in subscribers' hands within a week of the editorial deadline. And corrections, updates, or alternative viewpoints can be out two or three weeks later.

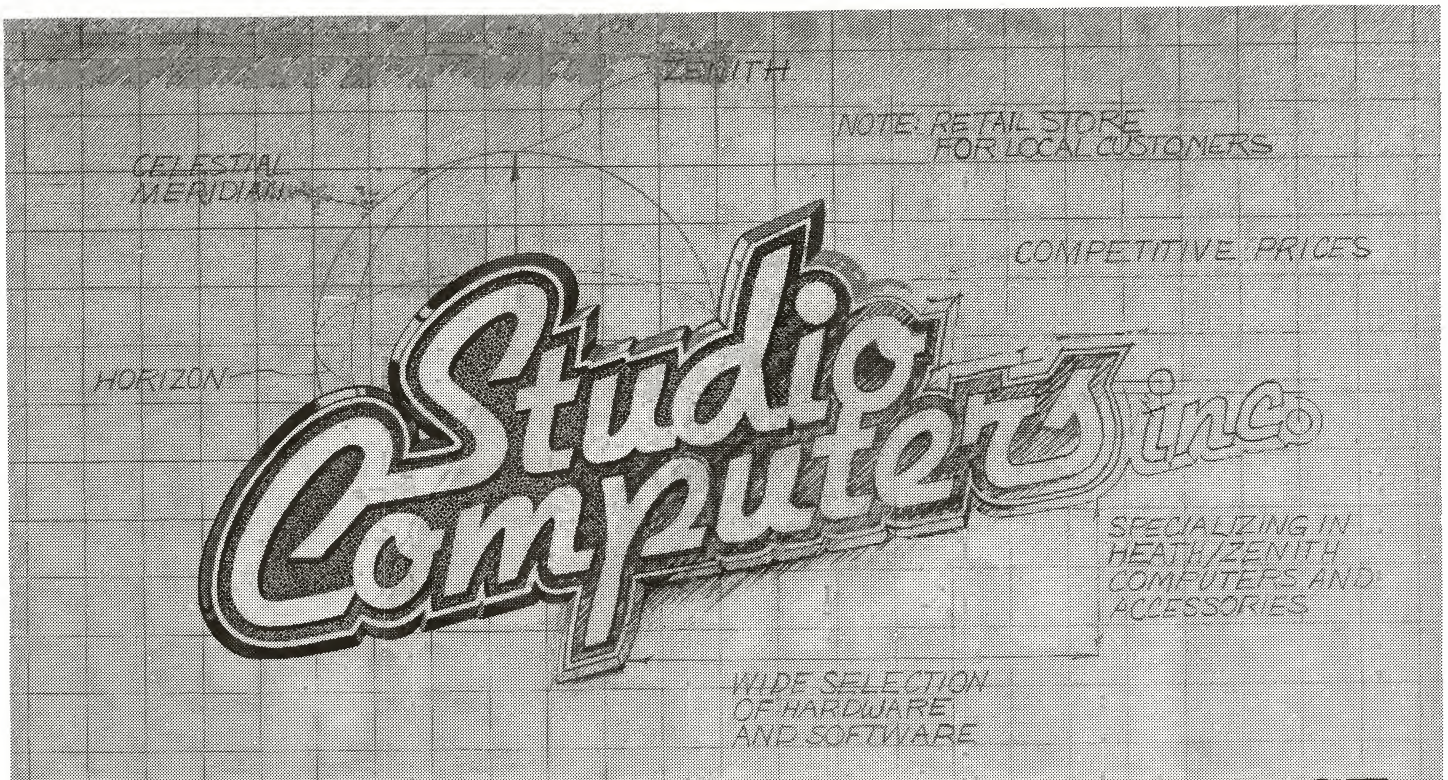
Buss was started seven years ago, a few months before Heath's first shipments of the H8 computer. So I figure I've been supporting Heath/Zenith computer users longer than anyone at the Heath Company or Zenith Data Systems. I intend to continue supporting you, whatever happens to them.

Buss has increased in frequency from six issues a year through nine, 16, and 20. Its current somewhat-flexible schedule seems appropriate for the forseeable future. *Sextant* has gone from four issues annually to six. That also seems appropriate for the next year or two. But we might want to make a change some time. So both *Buss* and *Sextant* subscriptions have always been sold by number of issues, not a fixed period of time. (Each subscriber's "last issue number" is shown at the right end of the top line of the address label.)

Several of our readers have requested that we give more attention to the information needs of novices. So this issue we're starting a new department: "Standard Operating Procedure." We hope it will also be valuable for those who have significant experience with computers, but may have overlooked a simple trick or two.

With issues coming closer together, we'll also be more willing to print a series of articles on the same subject. We've started work on two programming language series—one on Pascal and one on C. *Sextant* will continue to cover events in the Heath/Zenith community. With 50% more issues per year, we'll sometimes be able to do so in a more timely fashion. This issue we're happy to be able to bring you remarks of Terry Jensen which should be of interest to a far larger audience than just those who attended CHUGCON 83.

Charles F. Floto



Circle #170 on Reader Service Card

Letters

Z-BASIC Battleship

One thing I liked about your magazine is the shipping protection; having the magazine enclosed in a plastic bag is a great idea. I like the extra care you have taken to make sure that the magazine gets to my house in A1 shape. I hope the bag system is something you can keep doing.

I like having programs with BASIC listings; one suggestion I would like to make is that when you print a BASIC program, include a listing of the changes that would have to be made to make it work with Z-BASIC. The article with the program for Battleship in issue No. 7 looked really interesting, so I decided to rewrite this program to work on my system, without changing it any more than I had to. Instead of the direct cursor addressing used for the H19 I used the LOCATE command of Z-BASIC. Here is a table of some of the other changes:

CP/M	Function	Z-BASIC
CL$	Clear Screen	CLS
GR$	Graphics Mode	SCREEN 1,0
RG$	Exit Graphics	SCREEN 0,0
RV$	Reverse Video	SCREEN 0,1
VR$	Regular Video	SCREEN 0,0
BE$	Bell Code	BEEP

Keep up the good work. I will be looking forward to the next issue.

Dan Walls
Russell, KS

Recatalog Your HDOS Disks

The Fall 1983 issue of *Sextant* (No. 7) contained my article "Index Your HDOS Disks." Along with the MBASIC program was presented a series of patches that would allow the program to restart from a previously run catalog—in effect, to "pick up where it had left off." The advantage of that was not having to spend time re-cataloging disks that had already been indexed. The attached listing contains a second set of patches that will further extend the usefulness of the cataloging program by allowing it to delete a disk from the existing catalog so that it can be re-cataloged. This is handy when you have made several changes to the files on a disk and don't want to go to the trouble of rerunning the complete catalog.

It is possible to delete a certain disk's files from the catalog without these patches. You can use an editor such as PIE to delete the entries from the restart file SY0:FILENAME.DAT, but this is a pretty time-consuming process. Once these patches have been installed, it takes only several seconds to edit the master catalog from the MBASIC program.

To install these patches, you should first add the "restart" patches presented in Listing 4 of the article, then add the "edit" patches from Listing 1. Then you're all set. When running the program, it will inquire whether you want to restart from a previous catalog. If you answer "YES", it will load the catalog and then ask if you wish to purge the files from any disk so that that disk can be re-cataloged. If you answer "YES" it then prompts you for the name of the disk and purges all the files on that disk from the catalog. It will then ask for the name of the next disk and keep on merrily purging files until you simply hit RETURN. After that the program runs as before.

One caution: there is an array in the program that stores the names of the disks. This array, V$, is dimensioned for 256 names. When you delete a certain disk from the catalog and then re-catalog it, its name will be stored twice in V$. It is theoretically possible, after doing a lot of re-cataloging, to fill up V$ and have the program abort with an error message. If you expect to do a lot of re-cataloging then you might want to increase the dimensions of V$. The reason that the program doesn't purge obsolete names from V$ is that there is a lot of cross-referencing of this array to other arrays and many statements would have to be added. It's simpler to increase the dimensions of V$.

Stephen E. Auyer
Pittsfield, MA

```
1628 INPUT"Do you want to re-catalog a disk <YES/NO>";S$
1629 IF LEFT$(S$,1)="Y" OR LEFT$(S$,1)="y" THEN GOSUB 2940
2940 REM
2950 REM - routine to purge portions of data file.
2960 REM
2970 D$="": INPUT"Name of disk to be re-cataloged <RETURN to stop>";D$
2980 IF D$="" THEN RETURN
2990 DD=0
3000 DD=DD+1: IF V$(DD)=D$ THEN 3030
3010 IF DD<=ND-1 THEN 3000
3020 PRINT "Cannot find disk ";D$: GOTO 2970
3030 FD=0: I=1
3040 IF FV(I)<>DD THEN 3080
3050 FD=FD+1: IF I>=NF-FD+1 THEN 3080
3060 FOR J=1 TO NF-FD: FV(J)=FV(J+1): FL(J)=FL(J+1)
3070 FI$(J)=FI$(J+1): DA!(J)=DA!(J+1): NEXT J: I=I-1
3080 I=I+1: IF I<=NF-FD THEN 3040
3090 NF=NF-FD: PRINT FD;"files were deleted.": GOTO 2970
3100 END
```

Listing 1: This patch enables the Disk Master Directory Program to delete a disk from the existing catalog so that it can be re-cataloged.

More on the Z100's Key Click

I enjoyed D. C. Shoemaker's article on Z-DOS assembly language in the Winter 1983 issue of *Sextant*. I don't have much experience in programming, and it was very interesting to see how .COM programs are developed. However, as his example he made a program for eliminating the key-click feature on the Z100, and suggested that this was the simplest way to "do something useful which would be cumbersome or impossible to do any other way."

As a matter of fact, there is a much simpler way to eliminate the key-click or, for that matter, to set any other desired Z100 parameters at start-up in Z-DOS (see Z100 Users Manual, pp. B.14–B.17 for a list of these parameters and their ESCape codes). Most full-screen editors have a "next character literal" function by which ESCape characters can be entered in a line of text. I am familiar with Vedit and Final Word, both of which have it. If you prepare your AUTOEXEC.BAT file with one of these editors, and have the first line of the file consist entirely of the desired escape codes (including ESCx2 to eliminate key-click), these will all be entered automatically at start-up, which will configure the Z100 exactly as you prefer. Try it. It works.

A. Kent Christensen
Ann Arbor, MI

Jumpers for 16K ROM

A note regarding the confusion on jumpers (Glenn Roberts' letter, Winter '84) when a Z100 has the 16K "fancy" monitor ROM installed. Existing ROMs out there in the world should fall into one of the following categories:

Label	Version*	Size (bytes)
444–87	1.0	8K
444–87–1	1.1	8K
444–87–2	1.2	8K
444–87–4	2.3	16K
444–87–5	2.5	16K

*type 'V' at hand prompt

A board with an 8K ROM should have both jumpers J101 and J102 in the 0 position (closer to the center of the motherboard). With a 16K ROM, jumper J102 should be moved over to the 1

position (toward the edge of the board). This is an area for possible confusion because of a screening mistake by the PC board manufacturer. Jumper J102 is a make-or-break for Buffered Address 13; and J101 does likewise for BA14. This is probably the reverse of what your Motherboard Schematic MB2 tells you about these two jumpers (at least, it is so on my 1982-vintage schematic).

It takes eons for the board supplier to implement any design changes or corrections; but, hopefully, both the screening error and the blue wires on the back of the board will disappear—as the feller said—Real Soon Now. We hope that this helps.

Mark R. Van Sickle, Mark Mangerson
Z100 Motherboard Techs
Circuit Board Test Dept.
Heath Company
Benton Harbor, MI

No Response from RP Enterprises

RE: R. L. Peterson DBS RP Enterprises

On or about June 15, 1983, after the article in *Sextant* regarding the above firm's product, I sent an order and check for their product.

On or about August 22, 1983, I requested information on my undelivered order.

About a month ago [in mid-October] I wrote cancelling my order and requesting return of my check. The check, fortunately, has not been cashed.

None of these communications has been acknowledged.

In the Fall issue of *Sextant* you carried an advertisement for this firm. Since you have promoted this firm via article and advertising I think you owe it to your subscribers via *Buss* and *Sextant* to advise them not to deal with this unreliable and non-responsive firm.

You might also mention that a similar product is available from Analytical Products, who advertised on page 90 of the Fall *Sextant.*

Don M. Deck
Lone Pine, CA

[*Analytical Products, 29924 Road 168, Visalia, CA 93291, 209/747–3235.*]

Wild Interrupts

I own an H120 computer and am contemplating the purchase of another. It was my understanding from the salesman that the Heath 120 was compatible with the IBM PC. I preferred buying the Heath but liked the idea of using the many programs available for the IBM. I have tried to run VisiCalc as well as several others that were designed for the IBM but they would not run. All I got was a wild interrupt.

The dealer now advises me that I need an IB Emulater to run IBM programs, and the execution time will be increased. He also advises me that not all IBM programs will run, either.

Can you advise me as to a good solution to this problem, and does Zenith or Heath plan a hardware modification in the near future that will allow compatibility with IBM software in the future?

My decision to purchase the second unit will be based on a solution to that problem.

Charles F. Williams, D.D.S.
Jacksonville, FL

For an explanation of the causes and some cures for Z100/IBM incompatibility, see "Why Won't All That IBM-PC Software Run on My Z100?", by Graham Wideman, in this issue. We will be dealing with the area of software compatibility in future issues.

Lucidata Pascal

I have read and enjoyed Mr. Kenner's article in *Sextant,* issue #6, "Pascal Elucidated," devoted almost exclusively to Polybytes' Lucidata Pascal compiler, and have some observations concerning his follow-up letter in issue #7, "Lucidata Pascal Addenda."

I've run a comparison between RUNCOM and PRUN on several of my programs, plus the PRUN configure program (CONFIGP.PAS) included with the Polybytes Lucidata Pascal compiler, and have consolidated the results for anyone interested. (See Table 1.)

These results were obtained using a double-sided 80-track drive and HUG's system driver. As you can see, the time to compile a typical program with a 56K system is almost three times faster using RUNCOM rather than PRUN.

Mr. Kenner, in his article, mentioned a bug he ran across during his introduction to the compiler but never mentioned what that bug was. One runs up against one bug when one tries to compile CONFIGP.PAS, the configuration program supplied with the compiler. It seems that hex addresses cannot be used as constants. Editing CONFIGP.PAS and substituting the hex addresses in the program fixes that bug.

Another bug which I found and which was addressed by Mr. Reeve was my inability to use dual pragmats in the PRUN command line, such as $B for line input to a program combined with $G for a stand alone program.

DEV:PRUN $B $G:PROGRAM PROGRAM

will not work as one would suppose. Mr. Reeve has informed me that

DEV:PRUN BG:PROGRAM PROGRAM

will, although I have not tried it, preferring to set LMODE within the program such that both character and line input mode can be used when appropriate.

Donald E. Risher
Charleston, WV

Program	.PAS Sectors	.BIN Sectors	COM Bytes	Compiler Time RUNCOM	Compiler Time PRUN	Ratio RUNCOM/PRUN
LISTDIR.PAS	11	6	1500	1:28	3:54	.378
CORRECT.PAS	13	5	1104	1:36	4:13	.378
MAIL.PAS	26	12	3024	2:39	7:37	.349
CHECKING.PAS	41	19	4804	4:12	12:48	.327
CONFIGP.PAS	50	31	7656	5:47	17:13	.336

Table 1: A comparison of compiler times for RUNCOM and PRUN.

Corrections

Some backslashes were omitted from the article "Letter-Quality Printing on a Dot-Matrix Budget," by Gordon W. Nichols, in the Winter 1984 issue of *Sextant.* In the second column of page 21, the command to underline the word "computer" should be: "\OUT25\computer\OUT 25\".

Applied Statistics, mentioned on p. 87 of the "Z100 Software Roundup" in *Sextant* #7, is available from Sunflower Software, 13915 Midland Drive, Shawnee, KS 66216, 913/631–1333.

Sextant readers interested in Public Domain Software from Furman Smith (Games Roundup, p. 50, *Sextant* #8) may obtain it directly from him at 87 Creek Drive, Montgomery, AL 36117.

A Z100 on Every Desk: Clarkson College

ZDS stumbled into the academic marketplace when one college ordered 4,000 systems.

Frederick Zimmerman

Photo Courtesy of Zenith Data Systems

Clarkson College of Technology is the first college to issue a personal computer to all its entering students. Every new student in Clarkson's class of 1987 has been issued the low-profile version of the Z100, with one disk drive and 192 kilobytes of random-access memory (RAM). The students each pay slightly more than $200 per semester, plus a one-time maintenance deposit of $200, for a total outlay of about $1800—and they get to keep the computer after they graduate.

Clarkson is a 4,000-student coeducational engineering, science, and management school located in Potsdam, New York, in the state's northernmost tip. There are already about a thousand Z100s on the Clarkson campus, and the college will be adding another thousand a year with each new entering class. Eventually, every student and faculty member at Clarkson will have a Z100.

Clarkson's activities are likely to be of interest to Heath/Zenith users for several reasons. First, Clarkson is going to be one of the first educational institutions where microcomputers will be issued to every student. The widespread availability of microcomputers may be a very important development in education—and they're using *our* Z100s.

Second, selling so many computers to Clarkson has helped Zenith Data Systems (ZDS) sell Z100s to other academic institutions. ZDS has now made group purchase arrangements at more than 75 campuses. Zenith is establishing a niche for itself in the academic computer market.

Third, the national news coverage resulting from the Clarkson sale has provided some important publicity for Zenith Data Systems. ZDS has an unusual and perhaps enviable relationship with its customers: their biggest complaint is that Zenith doesn't sufficiently advertise the virtues of its products. When Clarkson issued its Z100s, they appeared on ABC's "Good Morning, America," as well as the nightly network news shows, and the AP wire carried the story.

Finally, there is a higher concentration of Z100 users at Clarkson than virtually anywhere else in the world—perhaps even higher than in Benton Harbor-Saint Joseph, Michigan, the site of the Heath/Zenith manufacturing facility. Since Clarkson is primarily an engineering school, there will be a substantial number of technologically sophisticated people playing around with the Z100. Clarkson could become a major source of new software and hardware products for the Z100.

In late September, four weeks after the beginning of classes, I spent three days visiting Clarkson.

How Clarkson picked Zenith

How did Clarkson get hooked up with Zenith? I asked David Bray, Dean of Educational Computing Systems.

"Well, it was a chance occurrence, really. When did we start? Two years ago [1981] we got started on this plan" to equip every student with a personal computer. "We got serious in October. In the spring [of 1982] we had a small committee—I hate to call it a committee, it was really a class. We invited faculty members, and we actually signed students up, and they got credit for attending class. We did all kinds of things concerned with what kind of micro we should pick, what kind of network we should work on....

"We'd settled that we really needed an IBM PC-type compatible machine, compatible in the sense of reading disks—just to solve the disk problem. Because it was just clear that so much software was going to be made available for the 8086 and 8088 processors, and a lot of colleges were interested in the IBM. So it was pretty clear that we just had to do that. We made some very informal specs, and we sent them to anyone we

knew that was making a 16-bit machine, that we thought would be IBM-compatible.

"In the end we heard—seriously heard—from Northstar, IBM, DEC—and a lot of other people"; Bray noted Commodore in particular. "So we heard from quite a few people. But right at the end, when we were about to make a decision, on the first of August, we got a call from the local Potsdam dealer. He said, 'We hear you're looking for lots of computers. Maybe we can help.' I said, 'Well, I doubt it, because we're working directly with the manufacturers. We couldn't possibly deal with a dealer from a price point of view.' He said, 'Well, I know, but maybe I can help anyway.' So I said, 'Sure, fine,' so I told him all our specs on the phone.

"I hung up and laughed, told my secretary: 'There's another one of those nuts.' Because we got a lot of people who called, thinking they could somehow help. And by God, an hour later the national sales manager of Zenith called. He said, 'Hey, Mike Jones says you're looking for a 16-bit machine.' I said, 'Yeah, I didn't know you guys were making one.'

"He said, 'Well, we haven't announced it yet, but we are, and we're about to announce it any day.' So I said to myself, 'Oh, yeah, another two years and they'll have it.' So I told him the specs, and I said 'When can we have a prototype?' I didn't really care about the prototype, I wanted to know where the production situation was. He said, 'Oh, next week.' I said, 'Oh, yeah?' 'Yeah, we're starting production next week.'

"So I told him all the situation, told him the kind of pricing we needed, all the software we needed. He said, 'Well, I'll let you know.' Twenty-two hours later, he called up and said, 'Okay, we'll do everything you want.' We said, 'Great, get us that machine there quick.' Our leading contender probably was Northstar. But we weren't awfully happy with that in the sense that they would have had the Advantage this year, and next year they're going to have something new.... So then the machine [the Z100] came, and it took us five minutes to decide that was it."

Zenith supporting Clarkson

Zenith Data Systems' top management clearly regards Clarkson as an extraordinarily valuable customer. For example, both Zenith and Clarkson were concerned to make sure that there would be no defective units issued when the freshmen received their Z100s. To minimize damage caused by shipping and handling, Zenith sent the first batch of 900 Z100s on three ZDS trailer-trucks. A crew of three service experts was sent along to check each machine after unloading.

Dean Bray told *Sextant*: "They took every one of those machines out of the box—900—and they ran them through extensive tests, and it turned out they had 3% failure, which is exactly what you'd expect. I'm an electrical engineer, and I know that no matter whether you're building a toaster or a television set, and everything's perfect, you'll have 3% failure in the system. That's the absolute standard rate on anything, no matter what it is: resistors, anything. And by God, they got exactly three percent."

When the first batch of computers was distributed August 25-26, Don Moffet, president of ZDS, was there to give an opening address to the entering class. "The president of the company spent two days here.... The national service manager came in, spent two days here. It's unbelievable, just tremendous support. You can sure quote me on that," said Bray. "I've never seen anything like it."

"We've been working with them ever since, and they've just been unbelievably fantastic to work with.... I've never seen anything like it. We could call them up, and anything we want, they'd say 'fine.' ... We worked for four months on Digital and IBM and Burroughs and Northstar and Apple, with their Macintosh, and we just saw nothing like the Zenith response."

"IBM FORTRAN ... is unusable"

Some things Zenith has done to support Clarkson may end up benefitting other Zenith users. For example, Bray told me: "I think we're somewhat responsible for having a FORTRAN that really works on the Z100.... The IBM FORTRAN ... is unusable, *absolutely* unusable.... [Zenith] just pushed Microsoft like crazy.... They made Microsoft make [Zenith FORTRAN] pass all the National Bureau of Standards" tests. (One rumor has it that IBM's FORTRAN didn't pass *any* of the NBS tests.)

Zenith is offering FORTRAN and Pascal to Clarkson students for an exceptionally generous price, a fraction of the normal cost of such professional-quality high-level languages. Incidentally, Clarkson has developed some FORTRAN and Pascal graphics subroutines to run on the Z100. These subroutine packages give students

Toolbox Utilities

Despite the fact that SETUP is really a rather trivial utility, I found it among the most entertaining programs in the Freshman Toolbox. SETUP is a menu-driven utility which allows you to choose new foreground and background colors for the screen; to choose between blinking and non-blinking block and underline cursors; to turn the key click off and on; and to control several other parameters of the Z100 display. (D. C. Shoemaker's article in *Sextant* #8, Winter 1984, describes a simple assembly language programming technique for accomplishing such tasks.) Moreover, with AUTOEXEC.BAT you can instruct the Z100 to automatically set itself to your favorite screen configuration each time you boot. All you have to do is add a command to your AUTOEXEC.BAT command file which calls SETUP's sister program DOSETUP.

The Freshman Toolbox also includes utility programs which enable you to dump the contents of the Z100 screen to various printers.

Finally, the Toolbox includes a pair of utilities which might save some time for those of you who normally use more than one printer. SETSER temporarily reconfigures Z–DOS for a serial printer: 1200 baud, no parity, one stop bit, XON/XOFF (DC1/DC3), no pad characters, no parity-bit stripping either on input or on output. SETPAR temporarily reconfigures Z–DOS for a parallel printer. Simply typing six letters—SETPAR or SETSER—is noticeably quicker than going through the rigamarole of CONFIGUR.

ZPILOT: a simple programming language for educational applications

Clarkson programmers have developed a Z–DOS version of PILOT, a simple programming language which was designed specifically to help non-technical users create interactive educational programs. As Clarkson's documentation explains, "PILOT is ideal for the instructor who wishes to write outlines, quiz his students, and provide guidance through interactive programming." PILOT is an acronym for Programmed Inquiry, Learning or Teaching. Clarkson's version of PILOT is known as ZPILOT, or "PILOT-Plus."

One of the reasons many educators find PILOT useful is that they can begin programming after learning only a few statements. Clarkson faculty and students have devised a number of ZPILOT programs to help students with the readings for their courses.

Students can copy the ZPILOT programs at a public-access Z100 in the Faculty of Liberal Studies building, and then run the software on their personal Z100s. The programs are designed to help the students test whether or not they have understood the weekly readings.

I tried several programs written for Professor John Serio's Humanities 195: "Great Ideas of Western Civilization." The quizzes covered such topics as Socrates, the Bible, Shakespeare, and *Zen and the Art of Motorcycle Maintenance*. I concluded that they are useful for a limited, specific purpose: making sure that the students have read the material to the instructor's satisfaction. The higher educational value of the programs depends largely upon the quality of the questions the instructor tells the computer to ask. If you have good questions to ask, ZPILOT will make it easy for you to ask them. And if you are treating a subject which is well suited to short-answer or multiple-choice quiz programs, ZPILOT may be the language for you.

graphics capabilities similar to those available under Z-BASIC.

Zenith's generosity may possibly be guided by a longer-term interest in Clarkson's potential to develop as a source of technical support for the Z100. Since every Clarkson student is required to take a programming course before graduation, and such courses will likely use either FORTRAN or Pascal, there will soon be 4,000 students at Clarkson who have had some formal training in FORTRAN or Pascal programming *on the Z100*. If even a few of those students eventually become skilled enough to develop salable programs, Clarkson could be a fertile seedbed for Z100 software.

Current software development at Clarkson

Indeed, there is already some interesting Z100 software under development at Clarkson.

The Freshman Toolbox, which was issued to entering students along with each Z100, includes some workhorse programs, such as Clarkson's standard word processor, GALAHAD. In addition, the Toolbox provides students with various utility programs which will facilitate the use of the Z100 and the peripherals available on campus (printers, etc.). Finally, The Freshman Toolbox includes some programs specifically designed for the educational environment: PROOF, a spelling-checker program with features which will force students to build their vocabularies; and KP.BAS, a BASIC program designed to improve students' touch-typing skills.

Other software is being developed for specialized audiences at Clarkson. Members of the Faculty of Liberal Studies needed an easy-to-learn language, so Clarkson has developed a Z–DOS version of PILOT, a programming language which is specifically designed to make it easy to write interactive educational programs. On the other hand, computer scientists and advanced programmers like fast, powerful languages such as C, so work is underway on a Z–DOS C compiler.

(Further information about some of the programs mentioned in the preceding paragraphs may be found in the accompanying comments.)

I came away from Clarkson convinced that the college will indeed emerge as an independent source of Z100 software.

What's that microcomputer doing on my desk?

Despite the fact that Clarkson is a "College of Technology," not everyone on campus was happy to learn that microcomputers were about to become part of the daily life of every student entering Clarkson.

In November 1982, the Clarkson administration first announced the plan to issue Z100s to members of the class of 1987. The reaction from students was mixed. Later that month, Clarkson's student newspaper—*The Integrator*—published the results of an opinion survey.

(Since the survey was conducted in 1982, it couldn't include anyone from the class of '87. Moreover, almost 90% of the respondents were members of the classes of '85 and '86. Apparently, members of the classes of '83 and '84 simply felt less interest in the survey, knowing that they would have little or nothing to do with the Z100s.)

There were some grumblings that the administration hadn't given students much voice in a major decision about the college's future. "Why doesn't Clarkson give the students a chance to voice their opinions before they make the decisions?" The survey comments also indicated that there were misgivings about the administration's decision to make the purchase mandatory, even at the discount price of about $200/semester plus a $200 one-time maintenance deposit. "Make it by free will," one respondent suggested.

There was a considerable degree of pessimism as to the likely social effects of having the Z100s on campus. "I seriously wonder what kind of vegetables this program will attract." A more optimistic prediction: "At first the project will attract a new type of high school student, the 'digithead,' but in time, the Clarkson student will return to his original form."

One of the survey questions was: "How do you think the Z100 will affect education at Clarkson?" Of course, some students were optimistic: "I think it will greatly increase the value of Clarkson's educational system. Computers are here to stay." Some students were more

PROOF: a spelling checker that isn't a crutch

The Freshman Toolbox issued with each Z100 includes a spelling checker called PROOF, which is specifically designed for educational use. PROOF is designed to neutralize the danger that using a spelling checker might cause students' vocabulary skills to atrophy. (In much the same way that hand calculators have affected students' computational skills.)

PROOF comes with an expandable dictionary, or "lexicon," which initially contains only 2,500 words. This forces the student to regularly add new words to the lexicon. And the documentation warns, "Be sure the words you add are spelled correctly. There is currently no way to remove misspelled words once they are in the lexicon." This design feature obliges students to look up words which they are unsure how to spell.

Although PROOF is still under development, I should note that the version I saw has some irksome peculiarities. For one, it doesn't distinguish between upper- and lowercase. For another, when it checks the words in your text file, it strips apostrophes and hyphens. So if it finds an unfamiliar word like "Fred's", it will ask you whether you want to enter "freds" into the dictionary. When you're rounding up misspellings and typos, misplaced apostrophes and hyphens are among the usual suspects. Since PROOF isn't very good at spotting such errors, its value is rather limited for business and professional purposes. However, the program is very well suited for its intended purpose, developing students' vocabulary skills. If PROOF becomes available to the general public, parents whose children are using the family Z100 may find this program a valuable educational tool.

skeptical: "It will look good but do nothing," said one.

One student put it all in perspective: "it [the microcomputer] is only a tool for people to express ideas."

Will the Z100s improve students' writing skills?

The Z100s had only been on campus for a month or so when I visited, so it's hard to assess how the widespread availability of microcomputers will affect the quality of education at Clarkson. The most obvious effect I saw was that word-processing programs will relieve students of the burdensome necessity of retyping papers, saving them significant amounts of keyboarding time. The time saved can be spent on additional research, for example, or on a lot of rewriting.

When a student submits a paper, Humanities Professor John Serio returns it with comments, and asks that the paper be revised accordingly. Now, Serio says, he can ask students to do multiple revisions "without feeling like an ogre." He knows that all his students have word processors and spelling checkers, so there's much less reason for him to accept stylistic infelicities or grammatical irregularities.

When students finish their revisions, they visit Serio's office, bringing along the disk as well as the paper, and Serio pops the disk into his Z100 running the Clarkson word processor, GALAHAD. Thus he can demonstrate how the student should change the structure of an argument. He can actually use GALAHAD's "cut and paste" features to rearrange paragraphs within the paper. This procedure observes a cardinal rule of teaching (and entertainment): "Don't tell 'em—*show* 'em."

Serio is also asking his students to keep an "electronic journal." As soon as they finish a reading assignment, they're supposed to record their first reactions to the material. This serves two useful purposes: the act of writing forces students to think about what they read, and it gets them into the habit of sitting down at the keyboard.

Of course, asking students to keep a journal isn't a new idea. However, I think the presence of the Z100s at Clarkson should make it much more likely that the students will actually do the work. The difference between using a typewriter and using a word processor is almost like the difference between cooking dinner for yourself and dining in a restaurant every evening. Future generations of students will be able to approach writing without having to worry about all the niggling frustrations of typewriter repairs and liquid paper.

Clarkson supporting Clarkson

So far, everyone agrees that the Z100s have been integrated into campus life with many fewer frustrations than were expected. On the day the Z100s were distributed, a service fraternity called Student Orientation Service (SOS) helped the new students get the Z100s safely to their rooms. Juniors and seniors were stationed around the various dorms to provide help in setting up the Z100s. Faculty and administration gave orientation lectures. As a result of all this preparation, the Z100s' first day at Clarkson went virtually without a hitch.

The Clarkson administration has done its part to ensure that students get adequate "human factors" support for their new microcomputers after the first day's hoopla dies down. Faculty have been instructed to make freshman courses a bit slower this semester, to ensure that no one gets too far behind because of difficulties in adjusting to the Z100.

Clarkson is also providing maintenance support for the Z100s, based on the $200 maintenance deposit received from students when the Z100s were issued. Thus, if anything goes wrong, the student need not

C compiler for Z-DOS

According to Dean Bray, Clarkson is developing a Z-DOS compiler for C, a programming language which is a favorite of systems and applications programmers. Bray told me that the Clarkson compiler is based on the public-domain C compilers published by Ron Cain and Jim Hendrix in *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia* in May 1980 and December 1982, respectively. It seems likely that the compiler will be available by the time you read this. "We're going to have a neat C for the college environment," said Dean Bray. "That's going to be a very nice package for the computer science people."

worry about finding the cash to pay for repairs. Clarkson's electronics shop, which is now an authorized Zenith service center, will repair the machine, then subtract the appropriate amount from the student's account with the bursar.

Servicing nearly a thousand Z100s

I thought Clarkson's experience with a sample of almost a thousand Z100s might provide some helpful information for other Z100 owners. So I interviewed Bob Dwyer, who is the chief technician at Clarkson's Z100 service center, and asked him about the Z100s' service record after four weeks on campus. Dwyer gave the impression that the new users at Clarkson have done a good job of adjusting to their new computers. Overall, he said, "call-backs have been very low"—few repairs have been required.

> ***Now, Serio says, he can ask students to do multiple revisions "without feeling like an ogre."***

Nevertheless, I asked Dwyer to identify the most common problem he has encountered with servicing Clarkson's Z100s. He replied instantly: "cockpit error." In his experience, the most likely cause of a seeming malfunction is an underinformed user, not problems with the hardware.

When pressed, Dwyer admitted that he has found hardware faults in a few Z100s. "Disk drives are probably our biggest problem, but they're usually repairable." He offered one useful tip, gained from taking nearly a thousand new Z100s out of the box for testing. After a Z100 has suffered the rigors of being shipped cross-country, the disk drive controller board has a tendency to jiggle loose. This causes a "Device Error" message when the user tries to boot. Fortunately, Dwyer has a simple solution: "Pop the board out, then pop it back in."

Z100s networking with the Clarkson mainframe

How is Clarkson integrating a large number of newly acquired microcomputers into its existing mainframe data-processing environment?

Right now, there is a limited network installed in the Science Center, with 30 Z100s wired up to the school's IBM 4341 mainframe. These Z100s are open to public access, so students can bring disks to the Science Center and access the mainframe from there. When finished with work on the mainframe, they can take the disks home and continue work on a stand-alone Z100.

Clarkson's systems programmers have developed a number of programs to make it possible to use the Z100 and the 4341 in concert. For example, I saw a program called TERM, developed at Clarkson, which enables the Z100 to emulate an IBM 3270 terminal. TERM can also transfer files back and forth between a Z100 and the 4341. Similarly, there is software under development which will allow a Z100 to emulate a Tektronix 4010 graphics terminal.

One area of mainframe-microcomputer interaction will be word processing. GALAHAD, the Z100 word

Christopher Cleveland

I met some impressive students at Clarkson; one was a sophomore named Christopher Cleveland who's involved in a remarkable range of activities.

He's written "quick and dirty" Z100 versions of the games Breakout and Mad Bomber, as well as a graphics version of Adventure. He's also been working on a program to emulate Digital Equipment Corporation's popular VT100 terminal.

But even without that work, Cleveland would be keeping busy. For example, he runs a technical assistance clinic on the third floor of Clarkson's computer engineering building. His primary job there is to advise Z100 users (faculty and students) on how to interface their Z100s with peripherals, particularly printers.

As a personal project, Cleveland is working to develop a "game card" for the Z100. The S100-compatible card would have a sound-generator chip, a joystick interface, and a real-time clock.

Cleveland explained his design philosophy: "I'm primarily concerned with speed." To speed things up, he'll go so far as to "fly blind," without a monitor. "Sometimes I turn the computer on, tell it what to do, come back in 20 minutes, and turn the monitor on." This preoccupation with speed applies to Cleveland's academic career as well as his computing activities. He's a sophomore, but is on an accelerated program whereby he will graduate at the end of the '83-'84 academic year.

processor developed at Clarkson, will be sufficient for most users' purposes. However, Clarkson programmers have developed means to send data back and forth between a Z100 and the mainframe's word processor, a program called SCRIPT. If a Z100 user uploads files to the mainframe, SCRIPT can be used to perform sophisticated or unusual editing, text formatting, and printing operations.

Clarkson has ambitious plans for its future networking capabilities. "Down the road, we're looking for a true comprehensive data network, maybe even integrated with video and audio," said Richard Valente, director of the college's Educational Resource Center. (The ERC includes both the college's mainframe computers and the library.) The college will be following the development of networking capabilities and protocols very closely, with an eye towards Valente's ultimate goal: to make it possible to connect "every dorm room and every office" at Clarkson.

Future of the Z100s at Clarkson

When asked whether the Z100s made any difference in their decision to attend Clarkson, most of Clarkson's entering freshmen said that it was "a big plus," but not decisive. Some Clarkson students worried about the "digitheads," who will presumably continue to see the Z100s as "a big plus" for the next few years. But the likelihood is that fairly soon most colleges and universities will have some sort of group purchase program to make microcomputers easily available to students. Nevertheless, the distinction of being first remains. Clarkson has been a true pioneer in bringing *personal* computing to higher education.

A coup for Zenith Data Systems

The sale of Z100s to Clarkson was an important coup for Zenith Data Systems. The most obvious benefit, of course, was that the sale raised Zenith's profile in the academic microcomputing market. Zenith Data Systems now runs an ad in *The Chronicle of Higher Education* with the headline: "Meet Dean Bray and the roommate he picked for all of Clarkson's incoming freshmen." And many institutions of higher learning have now made group purchase arrangements with Zenith, including the University of Michigan and eight other Michigan state universities, Tufts University, and the Kansas state university system. Most of these arrangements were made after Clarkson announced its program.

Has the Clarkson contract helped Zenith sell Z100s? "There's no doubt in the world" that it did, Zenith Data Systems Marketing Manager Andrew Czernek told *Sextant* in a telephone interview. Making the sale "gave us a lot of important information on important marketing aspects of putting microcomputers in colleges and universities." Furthermore, and perhaps most importantly, the Clarkson sale "established the credibility of the Z100" in the academic microcomputing market.

The sale may also benefit ZDS in some more subtle ways. For example, recent Clarkson graduates are working for companies such as Texas Instruments, RCA, Hughes Aircraft, Grumman Aerospace, Eastman Kodak, Digital Equipment Corporation, and, yes, IBM. Each year from 1987 on, Clarkson College of Technology will graduate a thousand Z100 owners, most of them engineers. They will migrate into American business and American society—and they will bring along with them an enthusiasm for Zenith microcomputers.

Additional Information

Clarkson College
Potsdam, NY 13676
315/268-6400

Ordering Information

GALAHAD and ZPILOT are each separately available from Clarkson for $1,000 for a college-wide license, $49 for a single copy, and $29 each for quantities of 10 or more.

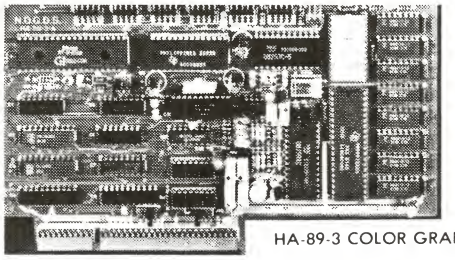
Graphics subroutines which give FORTRAN and Pascal the same color graphics capabilities as Z-BASIC are available for $1,000 for a college-wide license, $49 for a single copy, and $29 each for quantities of 10 or more.

A terminal emulation package which enables the Z100 to emulate VT52 and Tektronix 4010 terminals is available for $500 for a college-wide license. The screen-dump utilities are also included.

Clarkson software questions:
Dean David Bray
Clarkson College
Potsdam, NY 13676

H-8 and H/Z-89 ACCESSORIES

PUT COLOR GRAPHICS ON YOUR HEATH/ZENITH COMPUTER



HA-89-3 COLOR GRAPHICS BOARD

HA-89-3 COLOR GRAPHICS BOARD............................$299.00

This board will expand your H/Z-89 computer to include color graphics and sound effects with support for both HDOS and CP/M. High level application support under GRAPH-PAC-II (see description below), and all for a price much lower than other boards offering much less! Uses the advanced TMS-9918A Color Video Generator which displays brilliant color graphic output on any video device accepting NTSC signals. Included is a documentation package which contains a distribution disk with sample programs and driver routines, a 275-page manual with detailed information on installing and using the HA-89-3, a schematic, manufacturers' data sheets for the major components of the board, and a list of vendors who have commercial software packages available for this board. The standard board includes:

- Eight Analog/Digital Converter Channels
- Sound Generation for Special Effects
- 16-Bits of Parallel I/O
- Three Counter Timer Channels
- Priority Interrupt Controller

Optional Features Include:

- **Voice Synthesis using the VOTRAX SC-01A chip.......$130.00**
- **Am9511A Arithmetic Processor (floating point and trig functions).............................$175.00**
- **Two 12-Bit Digital/Analog Converters..............$110.00**
- **Joysticks (each) .... $ 14.00**

Add $15.00 extra if options are not purchased with basic board.)

Please specify operating system (HDOS or CP/M) and disk format (any standard Heath format).

HA-8-3 COLOR GRAPHICS BOARD............................$299.00

Create color graphics and sound effects on your H-8 computer. Supports HDOS and CP/M and high level applications with GRAPH-PAC-II (see description below). Uses the advanced TMS-9918A Color Video Generator which displays brilliant color graphic output on any video device accepting NTSC signals. The programmable sound generator makes a wide variety of sounds. Included is a documentation package which contains a distribution disk with sample programs and driver routines, a 250-page manual with detailed information on installing and using the HA-8-3, a schematic, manufacturers' data sheets for the major components of the board, and a list of vendors who have commercial software packages available for this board. The standard board includes:

- Eight Analog/Digital Converter Channels
- Sound Generation for Special Effects
- 16-Bits of Parallel I/O

Optional Features:

- **Am9511A Arithmetic Processor (floating point and trig functions)..............................$175.00**
- **Joysticks (each)................................. $ 14.00**

Please specify operating system (HDOS or CP/M) and disk format (any standard Heath format).

JOY-1 JOYSTICKS....................................(each) $ 14.00

For use with the HA-89-3 and HA-8-3 Color Graphics Boards described above. Please specify for which board you will be using it.

NEW FOR H-8 OWNERS

NOW YOU CAN INCORPORATE H/Z-89 I/O CARDS INTO YOUR H-8 SYSTEM!!!!!!

HA-8-89 BUS CONVERTER....................................$ 89.00

This card allows incorporation of H/Z-89 cards into your H-8 system with the following features:

- H/Z-89 card fits into H-8 cabinet
- One converter per H/Z-89 card
- Standard Heath ROM addressing

H-8 MUSIC SYNTHESIZER SYSTEM.............................$ 89.00

Expand your H-8 computer to include the latest pop hits! This board connects to your stereo system with two shielded cables (included). A true D/A converter provides excellent 27.5-3300 Hz frequency response, up to 9 harmonics. 4-part harmony in stereo. Enter any song, right from sheet music. Full documentation included. Requires min. 24K memory, H-17 floppy, terminal. Please specify operating system (HDOS or CP/M) and disk format (any standard Heath format).

OPTIONAL 8-VOICE MUSIC COMPILER..........................$ 39.00

Allows your system to produce 8-part harmony in stereo rather than 4-part described above. Requires 4 MHz Processor. May also be ordered separately for existing HA-8-2 boards.

GRAPH-PAC-II...$ 39.00

A software support package for the HA-89-3 and HA-8-3 Color Graphics Boards described above.

- Supports calls from:

  ASSEMBLY LANGUAGE (requires M-80)
  BASIC-80 (interpreter or compiler)
  TINY PASCAL
  PASCAL/MT+
  FORTRAN-80
  COBOL-80
  C/80 Ver. 2.0 (requires M80 and L80)

- Primitives and high level support
- Generates circles, lines, points, characters and bar graphs
- High speed object movement and collision detection
- User definable character fonts

Package includes your choice of one of the above languages. Additional languages may be ordered for $15.00 each. Please specify language(s), disk format and operating system.

PASCAL/MT+ for HDOS....................................$350.00

Contains six major components: compiler, linker, debugger, disassembler, librarian and distribution package. The recommended environment is 64K of memory (the Pascal/MT+ compiler will use the lower 8K for additional symbol table space), three 100K disks or two extended density disks.

- Supports entire ISO standard (DPS/7185)
- Includes variant records, sets, record files and procedure passing
- Modular compilation with external variables
- Overlays and chaining through programs
- Interrupt procedures and absolute variables
- Inline assembly code
- Generates native Z80 or 8080 Code.

Available in any Heath format, please specify.

NEW ORLEANS GENERAL DATA SERVICES

Ordering Information: Products listed are available from NOGDS, 7230 Chadbourne Drive, New Orleans, La. 70126. Check, Money Order, VISA or MasterCard accepted. Phone orders call (504)241-9495. Add $5.00 shipping (VIA UPS unless otherwise specified).

GRAPHICS

H/Z-19 The Imaginator H/Z-89

$445.

ASSEMBLED
AND TESTED

Professional Graphics at Practical Prices

IN USE THROUGHOUT INDUSTRY, EDUCATION AND GOVERNMENT

- High Resolution 504 x 247
- Accessible through any high level language FORTRAN, Pascal, BASIC Etc.
- Onboard Microcomputer eliminates processing load on host.
- Source Code Available
- Rich Graphics Instruction Set
- Mix text and graphics
- Tektronix® 4010-4014 Compatibility Option with GIN Mode
- Comprehensive Documentation Includes numerous examples
- Fully buffered for Asynchronous Operation
- All original H/Z features remain intact
- Low Cost - $445. complete, Kits from $215.

Cleveland Codonics, Inc.
P.O. Box 45259 Cleveland, OH 44145 (216) 327-6405

Tektronix® - Registered Trade Mark of Tektronix Inc.

Circle #112 on Reader Service Card

Drawing on Your Heath/Zenith Computer: Ed-A-Sketch

Ed-A-Sketch creates and saves attractive screen displays using the Heath/Zenith graphics character set.

Lee C. Syer

Have you ever wanted to create a graphics display or an attention-getting message on your computer? Usually, it takes many hours to construct things like this from scratch using BASIC or assembly language. They are difficult to produce at best.

Ed-A-Sketch, by Gail Halverson of The Software Toolworks, has changed all that. This graphics package makes light work of creating graphics pictures, screens for menus, extra-large letters for messages, and many otherwise time-consuming tasks involving screen control.

Photo Courtesy of Lee C. Syer

Figure 1: The alphabet file from Ed-A-Sketch. The letters forming "ED-A-SKETCH" were PICKed from the top and PUT at the bottom.

Ed-A-Sketch has many features that make it easy to work with. An especially nice one is the "second-chance" or OOPS! key. This would be a nice addition to any software package. More on the OOPS! key later.

The Ed-A-Sketch package has proven to be very useful to me, and I have enjoyed working with this program over the past few months. As I have become more accustomed to using it, I have learned that this program makes quick work of creating menus and graphics for use or display with nearly any type of programming format the user might need. I have been using Ed-A-Sketch to produce attractive and interesting menus for BASIC programs that are used in a small business application. It is, I feel, one of the most useful programs in my software library.

The Ed-A-Sketch package is designed for use with the H/Z89 or the H8 with an H/Z19 terminal. It comes ready to go under either the Heath` Disk Operating System (HDOS) with at least 32 kilobytes random-access memory (RAM), or CP/M with at least 40K RAM.

The program can be used to create images on the screen using the character set available with the '19 and '89: both letters and graphics characters, both regular and inverse video. This restriction to the Heath/Zenith character set is one of the few limitations of Ed-A-Sketch that I have found. Of course, the other side of the story is that it takes your terminal's graphics capability as far as it can without actually adding a graphics board.

Rectangular areas of the screen can be filled with any of the available characters. Lines can be drawn horizontally or vertically, and portions of any screen can be picked up and moved around. Whole CRT screens and portions of screens can be saved to disk for use with a choice of programming languages: Benton Harbor BASIC, Microsoft BASIC, assembler, or C/80. There's also a special Picture format for recall with Ed-A-Sketch or the HDOS TYPE command.

Set-up

The Ed-A-Sketch set-up procedure is very simple. After a backup copy is made, the SKETCH.ABS (which is the actual Ed-A-Sketch) program and several demonstration picture files are copied to a SYSGENed disk. The system is now ready to sketch. SKETCH is called up, and the operator is prompted to either sketch a new picture or recall an old picture file.

For starters, it is a good idea to use one of the demonstration files, which are already on the disk. These can be used for experimentation. The screen shown in Figure 1 is the ALPHABET file, which can be used during the example lesson to demonstrate the various aspects of the Ed-A-Sketch program. The larger letters have actually been constructed using the graphics characters from the Heath/Zenith system. The letters can be picked up individually or in groups and moved to different positions on the screen in regular or inverse video. In Figure 1, the "ED-A-SKETCH" shown below the alphabet was created by picking up the appropriate letters from above and moving them down to the bottom of the screen. Everything is done very conveniently using the cursor controls and the special function keys.

It is very exciting the first time a whole area of the screen is picked up and moved completely. You begin to realize just how easy this program will make your graphics computing from here on.

Cursor positioning

The cursor is moved around the screen by using the arrow function keys on the keyboard number pad. The TAB key functions as normal, with the ESC key causing a reverse of the TAB, or a backtab. The DELETE and BACKSPACE keys perform their normal functions, with an exception to be discussed later. The HOME key takes the cursor to the upper left corner from any position on the screen.

A shifted arrow moves the cursor to the end of the screen in the designated direction. The time for the cursor to traverse the screen without this last

Photo Courtesy of Lee C. Syer

Figure 2: This is a message that was incorporated into a BASIC program. The program flashed a welcome to some visiting friends.

feature can seem like forever. It is nice having what I will call the quick cursor. The cursor movements are very rapid and easy.

Defining areas of the screen

Large and small portions of the screen can be defined, filled (painted), or separated either as rectangles, squares, or lines. There is no provision for angular lines (e.g., 45 degrees) or circularly defined areas. This would be a nice feature, but I feel that for the price, this program does quite a bit. It would almost be asking too much.

The defining of areas is accomplished by first establishing a starting point by hitting the ENTER key on the number pad. An X appears on the screen in the cursor position, and remains there while the cursor is moved to define a rectangular area. These areas can be defined quickly and easily with the quick cursor described above.

The areas defined can then be manipulated in different ways including moving them around, filling them with graphics characters or letters, or saving the individual areas to a disk. If an area of the screen has a design or picture you want repeated somewhere else on the screen, you can pick it up and move it wherever desired using the same area-define function.

As I said, the large letters in the demonstration picture (the ALPHABET file) can be picked up individually and moved to other locations on the screen to spell out messages in large letters. The message in Figure 2 was designed quickly for some friends who visited recently. It was used in a BASIC program to demonstrate my computer. The creative possibilities for filling in or painting and moving areas of the screen around are nearly endless.

The OOPS! key can be quite handy.

IL, IC, DL, and DC

The keys on the number pad of the Heath/Zenith keyboard perform their stated functions: insert line (IL), insert character (IC), delete line (DL), and delete character (DC). They may also be used with the area-define function, for inserting and removing characters and spaces from the defined areas. Using these functions is one of the situations where the OOPS! key can be so handy.

If one of the delete features is used by mistake, the OOPS! key (which is actually the decimal-point [.] key on the number pad) will correct the error by restoring the screen to its previous condition. A warning though: this can be done only for the last keystroke made, and will not help you if you have made more than one improper keystroke. It has saved me on more than one occasion.

Painting with Ed-A-Sketch

All of the Heath/Zenith function keys are used in Ed-A-Sketch. The fill-to-right (f2 key), fill-from-left (f2 key struck twice), fill-to-bottom (f1 key), and the fill-from-top (f1 key struck twice) are all quick ways to paint or fill an entire area on one side of the cursor or the other.

This function works quite simply. If the cursor were positioned in the middle of the screen, the *fill-from-left* would fill the screen from the left side to the cursor position with the next printable character hit on the keyboard. This only affects the line the cursor is on. The *fill-to-bottom* function would fill the entire screen below the cursor with the next printable character or graphics symbol typed.

Again, the OOPS! key will function here and erase any painting which has been done by mistake. The painting keys also work with the area-define functions to paint those areas which have been defined with the X and the cursor.

There are two other keys which can be used in the painting mode: the ERASE key and the INVERT (f5) key. The ERASE key is used to erase the remainder of a line from the cursor to the end of the line, and will also erase an area defined by the area-define function. The INVERT key (f5) will cause the entire screen or a defined area to be displayed in inverse video. The INVERT function is a lot of fun to play with, as the negative image of the screen is displayed after this key is struck. I have designed a couple of creatures for a video display, and their whole disposition seems to change when shifted from normal to inverse video. (See Figure 3.)

Cutting and pasting

The Ed-A-Sketch functions also allow you to pick large and small sections of the screen as described earlier. The area to be picked is defined just as before with the ENTER key (an X is displayed on the screen) and the cursor is moved to define an area. After the area is defined, the PICK (red) key is struck and the defined area is stored in a buffer for future use. When the PUT (white) key is used, the stored area in the buffer is deposited on the screen with the upper left corner at the current cursor location. The PUT function may be repeated as often as needed. The information in the buffer is placed wherever you want it on the screen.

The defined area may also be saved to a disk and recalled to a specified location on the screen at a later date. This cut and paste function and the ability to

Photo Courtesy of Lee C. Syer

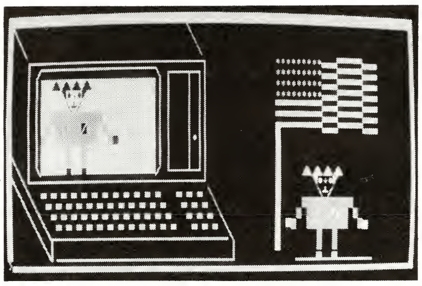


Figure 3: These are the creatures created with Ed-A-Sketch with a background created the same way. One creature is in regular video, the other in reverse video.

save a defined area are two of the most powerful features of Ed-A-Sketch. These features can be used to create slowly moving pictures with BASIC and assembler programs. The PICK and PUT functions are very useful in creating displays where areas are to be repeated. This is especially useful where detailed work has been done in one part of the screen and you do not want to waste time repeating that work. It is a simple and time-saving matter to PICK and PUT portions of the screen.

How to SAVE *work*

SAVE (the f4 key) is the link between the Ed-A-Sketch pictures which have been created and the disk on which they are saved. There are four formats for saving the pictures you have created: Benton Harbor BASIC; Microsoft BASIC; assembler; and a special Picture format which enables the picture to be recalled for viewing either as an Ed-A-Sketch picture or by using the TYPE function of HDOS.

It is important to note that pictures cannot be used again with Ed-A-Sketch unless they have been saved in the Picture mode. Therefore, if the sketcher wants to save the picture for future reference, it must be saved in Picture format first, recalled, and then translated to one of the other languages.

The assembler (ASM) files are stored in assembly-language format, in a series of "DB" statements, ending with a null character (0 byte). Programs saved in the ASM format can be used in assembly-language programs, or in C/80 programs using the #asm directive.

Both of the BASIC formats are similar. The picture files are stored as a series of print statements ending with a RETURN so pictures can be called as subroutines. The BASIC files all start at a specified line number (9000), and can be merged with programs in which the pictures are needed. This is a very nice aspect of the Ed-A-Sketch program, as it saves countless hours of determining escape sequences for the creation of graphics displays in your BASIC programs.

Photo Courtesy of Lee C. Syer

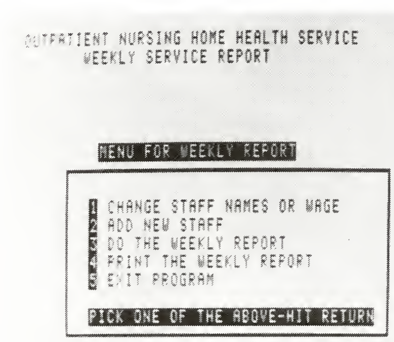


Figure 4: Menus such as this one are easy to create and modify as necessary using Ed-A-Sketch. They can be merged into BASIC programs after storage as BASIC statements.

The menus which I have designed for business programs have been put together in minutes using Ed-A-Sketch. They can then be saved for merging with programs at a later date. (See Figure 4.) The ease with which these menus and graphics are created and saved make the programmer's life much simpler.

Advanced features of Ed-A-Sketch

There are two features of Ed-A-Sketch which I have not found occasion to use except for experimentation. These are termed advanced features in the manual. I have experimented with both of them, and understand where they could be useful in some programming applications. The two features are as follows: the saving of small portions of the picture using *relative positioning* of the cursor; and the concept of *holes* and *blanks*.

The SAVE (f4) key may be used to save small portions of the screen to disk. These saved portions, however, must be positioned on the screen during recall from disk using relative positioning of the cursor. The normal full-screen pictures are all referenced to the top left corner of the screen. The smaller portions of a picture have no reference point and are stored in a format called Relative Positioning.

This means that the recalled portion of a picture must be given a reference point from which the display will start. This gives the programmer the option of positioning the recalled portion of the picture at any point on the screen simply by positioning the cursor at the point where the computer is to begin displaying that portion of the picture.

This feature can be used to create the illusion of slow motion on the screen by having a portion of the screen overwritten by the new picture portion. This feature gives the programmer a lot of flexibility in creating visual effects using "motion" on the screen.

The holes and blanks features refer to the actual state of a character position on the screen. The BACKSPACE and SPACE keys create blanks. Blanks, of course, are not printed, but are used in Ed-A-Sketch to create white or black backgrounds. A hole is created in the screen by DELETE and ERASE keys, and produces the apparent absence of any character.

This means that it is possible to position a picture or portion of a picture on top of a pre-existing picture on the screen. The picture underneath will show through the *holes* in the top picture, but will not show through the *blanks*. This feature of Ed-A-Sketch can be used to create overlays on the screen with only some portions of the underlying picture showing through. There is a special function key to show which areas of a picture are holes and which are blanks.

The holes and blanks can be used to create the appearance of a changing computer display, as in Figure 3. This would be done by positioning the printing or picture to be changed in a hole in the center of the larger picture, and

changing only the portion of the display inside the hole. This feature of Ed-A-Sketch has lots of possibilities.

The manual

The manual which accompanies the Ed-A-Sketch disk is sufficient. It is an 18-page manual with easy-to-follow instructions and examples. There are explanations of each of the different Ed-A-Sketch functions, and references are made to precautionary measures which must be taken while running the program.

After the introduction, there are instructions for beginners in the front of the manual. These instructions take the new sketcher through the rudiments of Ed-A-Sketch, going step by step. This guided tour is very helpful the first few times through, as there are quite a few keys and functions to learn. After the guided tour, there is a Reference Manual section which describes each key and function in depth. This section gives the working details on each of the special function keys on the Heath/Zenith keyboard. For example, the Reference Manual gives the following description of the BLUE key:

"GRAPHICS *The BLUE key toggles the graphics mode of the terminal.* If regular character keyboard input is in effect, the graphics character input mode is entered. If the graphics character input is in effect, regular input is entered."

I found the instructions generally clear and easy to understand.

The final page of the manual is a quick reference table of each key and the functions which it can perform. (See Table 1.) This table can be useful for refreshing your memory after the initial learning period.

The manual is adequate, but could be done with a little more polish. It does get you started quickly, and all the information you need is there. The material is presented in an easy-to-read style, and is understood quickly.

Key	Function Name	Function
F1	Fill To Bottom	*Fill to end of screen
F1-F1	Fill From Top	*Fill from the beginning of screen
F2	Fill To Right	*Fill to right end of line
F2-F2	Fill From Left	*Fill from left end of line
F3	Chart	Display graphics chart
F3-(.)	Erase Chart	Erase graphics chart
Enter F3	Display Holes	Toggle Display Blanks/Holes Mode
F4	Save	Save file
F5	Invert	*Inverse video (portions also)
Erase	Erase	*Erase (portions also)
Blue	Graphics	Toggle graphics mode
Red	Pick	Pick portion of screen
White	Put	Put picked area at cursor
ESC	Backtab	Back one tab position
IC	Insert Character	*Insert (#) characters
DC	Delete Character	*Delete (#) characters
IL	Insert Line	*Insert (#) lines
DL	Delete Line	*Delete (#) lines
Home		Home cursor
Up		Cursor up one line
Down		Cursor down one line
Right		Cursor right one position
Left		Cursor left one position
Shift-Up		Cursor to top of screen
Shift-Down		Cursor to bottom of screen
Shift-Right		Cursor to right edge
Shift-Left		Cursor to left edge
Enter		Enter define area mode (area)
Enter-Enter		Enter define area mode (line)
(.)	Oops	Cancel designated function
(0)	Inverse Video	Toggle inverse video mode

Note: Functions marked with an * may be applied to defined areas and may be cancelled by the Oops key. In the left column, the parentheses indicate keys on the numeric keypad.

Table 1: A quick reference for key functions. It is very handy after the rudiments of Ed-A-Sketch have been mastered.

Ed-A-Sketch overall

For the price of $29.95, Ed-A-Sketch is a real bargain. As I stated earlier, Ed-A-Sketch has become one of the most useful utilities in my software library. I have enjoyed using it and, most important, it has saved me hours of work. I feel that if a piece of software accomplishes both of the above, and can also be just plain fun, it is well worth recommending.

I have also used Video Artist from Newline Software, and find the programs comparable. Ed-A-Sketch has become my mainstay, however, because I used it first and have become more familiar with it. Both of them are fine programs and do an excellent job.

One of the few drawbacks to Ed-A-Sketch is that there is no provision at this time for printing the screen to a printer. Mention is made of this in the manual with a reference to possible use of the H/Z25, and the Epson MX–80 printers. The H/Z25 printer, as well as the Epson MX–80, should be capable of printing graphics from the screen. However, the '25 would need the addition of a new read only memory (ROM) chip in order to handle reverse video.

Another option is the 99G printer from Micro Peripherals, Inc. (M.P.I.). It should be able to handle all the graphics easily. However, you would have to buy or write a device driver for it to work with HDOS. Depending on the choice of printer, however, I think that most problems could be worked out by an enterprising user.

As I have stated, the only other real limitation is the lack of resolution imposed by reliance on the Heath/Zenith graphics characters; i.e., do not expect high-resolution graphics. And finally, there is the manual which I have described as adequate but not fantastic.

I do hope that Ms. Gail Halverson and the folks at The Software Toolworks can continue to produce such fine inexpensive software for the Heath/Zenith systems. I would like to make a suggestion to them, however: get to work on a universal OOPS! button. It would sell like hotcakes, because as long as there are computer programmers, somewhere, sometime, someone is saying "*Oops!!!!*"

Ordering Information

Ed-A-Sketch, $29.95
Specify disk format: 5″ HDOS or CP/M, hard- or soft-sectored; also standard 8″ single-density CP/M.
The Software Toolworks
15233 Ventura Boulevard
Suite 1118
Sherman Oaks, CA 91403
213/986-4885

Shakespeare had every gift for writing. Except one.

The FinalWord.

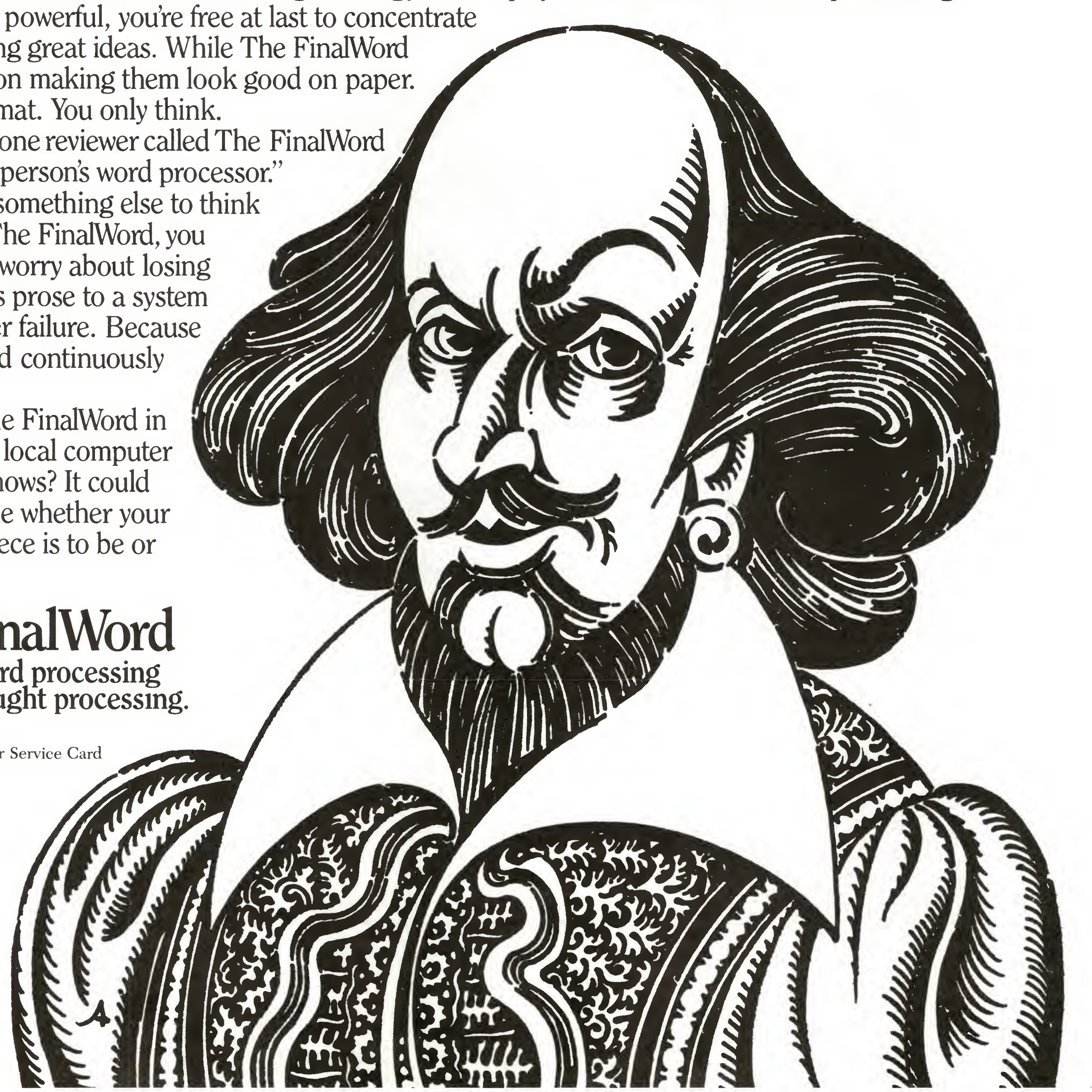
As the Bard himself would have said, "that which we call word processing will never be the same."

Because The FinalWord turns tough writing jobs into play. With an arsenal of word processing commands so powerful, you're free at last to concentrate solely on having great ideas. While The FinalWord concentrates on making them look good on paper. You never format. You only think.

In fact, one reviewer called The FinalWord "the thinking person's word processor."

Here's something else to think about. With The FinalWord, you never have to worry about losing your deathless prose to a system crash or power failure. Because The FinalWord continuously saves it.

See The FinalWord in action at your local computer store. Who knows? It could even determine whether your next masterpiece is to be or not to be.



The FinalWord

Finally, word processing becomes thought processing.

Circle #155 on Reader Service Card

Mark of the Unicorn
222 Third Street
Cambridge, MA 02142
(617) 576-2760

Optimizing Benton Harbor BASIC's Numerical Output

Understanding how B.H. BASIC stores numbers enables you to improve the precision and format of their display.

Charles W. Rogers

An early use many of us make of our Heath/Zenith computers is tracking financial items. Personal ledgers, expense accounts, net worth statements, and (of course) income tax records are very popular "first" applications. If we use Benton Harbor BASIC, we are likely to quickly confront two features which often needlessly discourage the budding enthusiast: B.H. BASIC does not provide a built-in mechanism to generate neatly aligned columns of numerical data. Furthermore, the number of significant digits of accuracy for arithmetic calculations is approximately 6.9.

These features do not impose serious barriers to the use of B.H. BASIC. We just need a routine to get around them. By understanding them, and through use of the procedures presented here, one can largely overcome these apparent limitations.

Illustrating the problem

To illustrate the features of Benton Harbor BASIC which we seek to work around is a simple matter. The program in Listing 1 can be prepared to print out two number series: the numbers between 9.5 and 10.1 in 0.1 increments; and those between 9,999.97 and 10,000.05 in 0.01 increments. This program produces the following output:

```
10 FOR X = 9.5 TO 10.1 STEP 0.1
20 PRINT TAB(5);X
30 NEXT X
40 FOR X = 9999.97 TO 10000.05 STEP 0.01
50 PRINT TAB(5);X
60 NEXT X
70 STOP
```

Listing 1: A Benton Harbor BASIC program to print out two number sequences: the numbers between 9.5 and 10.1 in 0.1 increments; and those between 9,999.97 and 10,000.05 in 0.01 increments.

```
9.5
9.6
9.7
9.8
9.9
10
10.1
9999.97
9999.98
9999.99
10000
10000
10000
10000
10000
10000.1
```

A close look at the output produced on your screen or printer will reveal that the first character printed in each row is in column six, rather than column five as might be expected from the TAB(5) in lines 20 and 50. Also, the first column available for more printing on each row will be the second position after the last character of the number. This is because BASIC generates a gratuitous space at the beginning and end of all numeric fields.

More significantly, the columnar presentation is not consistent with our esthetic traditions. We would prefer the decimal points to be in line—not the leftmost characters, as they are here.

Sometimes decimal points are present—sometimes they are not. Most significantly, look at the values printed for numbers over 10,000. BASIC can easily accommodate values up to 1 with 38 zeroes after it. However, the number of significant digits in arithmetic results is limited. Hence, 10,000.01, 10,000.02, 10,000.03, and 10,000.04 are all regarded as "10,000.0"; and the trailing zeroes are suppressed. And 10,000.05 is rounded up to 10,000.1.

The two features we observe, then, are: first, the number of significant digits in a result is limited; and second, the B.H. BASIC PRINT command is not directly usable for pleasing tabulations of columnar data. The first of these features, limited significant digits, is inherent to most implementations of BASIC or any other computer language. The limit can be changed by use of double-precision arithmetic in some versions of BASIC. However, there is always an upper limit. It is necessary for the programmer to know what the limit of precision is and work within the constraints. The second feature is easily accommodated by use of the algorithms presented later on.

Getting accuracy

Without going too deeply into the matter, we can still understand the significant digits issue. All numeric values are stored as "floating-point" numbers. Just as in newer calculators, there is no need to have the decimal point always occur at some fixed location. One byte is used for the number's "exponent." The exponent indicates where the "binary point" (as opposed to the decimal point) is to go.

The digits of the number itself, expressed in binary, are called the "mantissa." This uses three bytes. The first bit of the binary number, however, is a sign bit. Therefore, only 23 bits are available for the binary representation of the mantissa of the number.

These internal mechanics result in an upper limit of accuracy of six or seven decimal digits (depending on the value represented). For this reason, B.H. BASIC conservatively rounds all values to be printed to six decimal digits. The algorithms discussed later extract every last possible decimal digit of accuracy from the values. The guaranteed accuracy of the algorithms, however, cannot exceed the limits imposed by the mathematics of BASIC. Briefly, our upper limit will be 32,767.90 if we want

```
00100 OPEN "SY1:TEST01.DAT" FOR WRITE AS FILE #1
00110 PRINT #1, TAB(20);"SAMPLE RUN 1"
00120 PRINT #1,
00130 PRINT #1,
00140 Z=1.98
00150 GOSUB 9000:PRINT #1,
00160 Z=15
00170 GOSUB 9000:PRINT #1,
00180 FOR Z=23678.98 TO 33000 STEP 2000
00190 GOSUB 9000:PRINT #1,
00200 NEXT Z
00210 CLOSE #1
00220 STOP
09000 REM SUBROUTINE TO PRINT Z IN A DECIMAL ALIGNED FIELD.
09010 IF (ABS(Z)>32767.9)THEN PRINT #1,"OVERFLOW";:RETURN
09020 Z3=INT(ABS(Z)+.005)
09030 Z3$=MID$(STR$(Z3),2,LEN(STR$(Z3))-2)
09040 Z4=5-LEN(Z3$)
09050 FOR Z5=0TO Z4
09060 PRINT #1," ";
09070 NEXT Z5
09080 PRINT #1,Z3$;
09100 PRINT #1,".";
09110 Z4=INT((((ABS(Z)+.005)-Z3)*100))
09120 IF (Z4<10)THEN PRINT #1,"0";
09130 Z4$=MID$(STR$(Z4),2,LEN(STR$(Z4))-2)
09140 PRINT #1,Z4$;
09150 IF (Z<0)THEN PRINT #1,"-";:RETURN
09160 PRINT #1," ";:RETURN
```

Listing 2: A test program for the B.H. BASIC subroutine (shaded) to print numerical data in the range −32767.90 to +32767.90. Preceding zeroes will be suppressed and decimal points aligned.

```
    1.98
   15.00
23678.98
25678.98
27678.98
29678.98
31678.98
```

Figure 1: The output of Listing 2.

dollars and cents; or 8,388,607 for whole numbers. (A discussion of these limits accompanies this article.)

With a little thought, one can easily arrange meaningful displays of numeric information using only six or seven digits. For example, it is good standard accounting practice to omit fractions of dollars from our income tax returns. So, whereas a limit of $32,767.90 may seem restrictive in these inflationary times, many of us can do a credible job if the values are limited to $8,388,607. Again, these values are not arbitrary. They are the limits of guaranteed accuracy for the two subroutines presented later. Note that in both cases there are seven significant digits in the numbers.

Getting better appearance

Moving now to the question of neatly aligned columns of numbers, we should first note that the Heath Disk Operating System (HDOS) and especially B.H. BASIC are very fine, under-advertised, under-applauded systems. Furthermore, the price is right. If you purchased HDOS or the older cassette operating system from Heath, you got a version of "Extended Benton Harbor BASIC" at no additional cost. The storage requirements of competing software can make a one-floppy system look very tiny, indeed. With HDOS and B.H. BASIC, one can implement some quite extensive applications.

Among the niceties which some BASIC interpreters and compilers offer is a convenient mechanism to print columns of numbers with the decimal points aligned. This feature is usually implemented as PRINT USING. The proposed international standard for BASIC provides for PRINT USING approximately as follows:

PRINT USING "$##.##";2

would print $ 2.00. This is not available with B.H. BASIC.

Listing 2 is a subroutine which will print numerical data in the range −32767.90 to +32767.90, with preceding zeroes suppressed and decimal points aligned. The data are presented in the manner normally associated with dollars and cents reports. The routine could be modified to print larger values or to print more digits to the right of the decimal point. In these cases, the total number of significant digits would have to remain the same and the limit of accuracy would have to be re-established.

Listing 3 is a similar routine which prints values in the range −8,388,607 to +8,388,607 with commas inserted around the millions and thousands digits—in a manner pleasing to the eye and conventionally associated with numerical presentations.

To use either of these subroutines, simply set the variable Z equal to the value to be printed, then position the cursor or print head by issuing a "PRINT TAB(n);"—where n is the column for the start of the print field. Then call the appropriate routine. If the routines are used to generate files for later printing, then each PRINT command will have to be modified to an appropriate PRINT#chan command.

I hope the following explanations of Listing 2 and Listing 3 will enable you to understand the routines fully and also gain some insights into Benton Harbor BASIC.

Listing 2: Up to 32767.90

Line 9010 tests for the limit of guaranteed accuracy of the subroutine. It is always good programming practice to make such a test.

Line 9020 first rounds the value to be printed so that it will be accurate to the nearest hundredth. Then the integer portion of the value is assigned to the variable Z3. Z3 is the value which will be printed to the left of the decimal point.

Line 9030 removes the leading and trailing blanks from the ASCII representation of the value.

Lines 9040 through 9070 are a FOR...NEXT loop to print spaces in the beginning of the field so that when the character string representing the value of Z3 is finally printed, it will terminate in the position just to the left of the decimal point. An optional implementation to accomplish the same result would be successive tests of Z3 against 9, 99, 999, and 9999. This approach, as a matter of fact, appears to execute somewhat faster than the FOR...NEXT loop. It is the option used in Listing 3.

Line 9080 prints the character string built in line 9030.

Line 9100 prints the decimal point. For all values of Z accepted by line 9010, this will occur at the print position six spaces to the right of the cursor (or print-head) position when the subroutine was called.

Line 9110 rounds the value of Z as was done in line 9020. Then the integer portion of the number is stripped away by subtracting Z3. This leaves the fractional portion of the number to be printed. The program then isolates this fractional portion by multiplying it by 100. When the integer portion of the product is extracted, the result is a whole number which is correctly rounded to an integer corresponding to the fractional portion of Z.

Line 9120 prints a zero to the right of the decimal point for those values of the

```
00110 OPEN "SY1:TEST02.DAT" FOR WRITE AS FILE #1
00120 PRINT #1,TAB(20);"SAMPLE RUN # 2"
00130 PRINT #1,
00140 PRINT #1,
00150 Z=2
00160 GOSUB 10000
00170 PRINT #1,
00180 Z=1000
00190 GOSUB 10000
00200 PRINT #1,
00210 FOR Z=33 TO 8000000 STEP 1000000
00220 GOSUB 10000
00230 PRINT #1,
00240 NEXT Z
00250 CLOSE #1
00260 STOP
10000 REM SUBROUTINE TO PRINT #1,Z AS A COLUMNAR-ALIGNED INTEGER.
10010 IF (ABS(Z)>8388607)THEN PRINT #1,"OVERFLOW!!";:RETURN
10020 IF (Z=0)THEN PRINT #1,"        0";:RETURN
10030 IF (Z<0)THEN PRINT #1,"-";:GOTO 10050
10040 PRINT #1," ";
10050 Z1=ABS(Z)
10060 Z3=INT(Z1/1000):Z3=INT(Z3/1000)
10070 IF (Z3=0)THEN Z$=" ":PRINT #1,"  ";:GOTO 10100
10075 Z$="0"
10080 Z3$=MID$(STR$(Z3),2,1)
10090 PRINT #1,Z3$;",";
10100 Z2=INT((Z1/1000)-Z3*1000)
10110 GOSUB 10500
10120 IF (Z$=" ")THEN PRINT #1," ";:GOTO 10140
10130 PRINT #1,",";
10140 Z4=INT(Z1-(Z3*1000000)-(Z2*1000))
10150 Z2=Z4:GOSUB 10500
10160 RETURN
10500 IF (Z2=0)THEN PRINT #1,Z$;Z$;Z$;:RETURN
10510 Z2$=MID$(STR$(Z2),2,LEN(STR$(Z2))-2)
10520 IF (Z2>99)THEN PRINT #1,Z2$;:GOTO 10550
10530 IF (Z2>9)THEN PRINT #1,Z$;Z2$;:GOTO 10550
10540 PRINT #1,Z$;Z$;Z2$;
10550 Z$="0"
10560 RETURN
```

Listing 3: A test program for the B.H. BASIC subroutine (shaded) to print values in the range −8,388,607 to +8,388,607. Commas will be inserted around the millions and thousands digits.

fractional part in the range 00 through 09.

Line 9130 uses the same technique applied in line 9030 to strip away the leading and trailing spaces from the character representation of the number.

Line 9140 prints the fractional part of Z.

Line 9150 prints a minus sign to the right of the field for negative numbers. This minus sign could be printed preceding the field by placing this test before line 9050. The decision in this matter is purely personal. I selected the trailing minus because it corresponds to the conventions of an adding machine I use.

Line 9160 prints a space if the number Z was positive. This assures that on return to the caller the print head will always be nine spaces to the right of its position when the subroutine was entered. This is true for all values of Z: positive, negative, 0, or illegal values.

Listing 3: Up to 8,388,607

In some ways, this routine is less complex than Listing 2. It serves to illustrate that similar objectives can be achieved with alternate coding choices.

```
        2
    1,000
       33
1,000,033
2,000,033
3,000,033
4,000,033
5,000,033
6,000,033
7,000,033
```

Figure 2: The output of Listing 3.

Line 10010 tests for the limit of accuracy and returns the string of appropriate length for values which exceed the limit.

Line 10020 tests for 0 which is a special case. If this test were not made, the routine would print a blank line for zero.

Lines 10030 and 10040 put a minus sign (–) or a blank in the first position of the print field depending on the sign of the number.

Line 10050 sets up an assured positive number for all subsequent operations. An optional implementation here would be to round the value of Z (Z1 = ABS(Z) + 0.5). This, however, would change the limit of accuracy of the subroutine. As presented, the subroutine works correctly with integer numbers and truncates the fractional part of real numbers.

Line 10060 is a "trick" to force B.H. BASIC to generate a number which is the millions digit for values such as 1,999,999. Unless this two-step process is followed, B.H. BASIC will "round" the result of the division up, giving us "2" rather than "1" as the result. On completion of this line, Z3 is an integer value in the range 0, ..., 8.

Line 10070 tests for Z3=0. If this is so, then the number to be printed is less than 1,000,000 and we must print an appropriate number of spaces. The string variable Z$ is used to provide for either printing leading zeroes in subsequent fields (Z$="0") or for printing spaces in the position of leading zeroes in subsequent fields. If the value of Z3 is zero, then the number to be printed is less than 1,000,000. Therefore, there is at least one leading zero (and possibly more) to be suppressed. Z$ is set to the character space. Two spaces are printed. The branch to line 10100 then skips the next three lines.

Line 10075 is entered only if the number is 1,000,000 or larger. The string variable Z$ is set to the character "0" (zero). This variable is used to keep us from suppressing significant digits in the subsequent areas of the field.

Line 10080 is similar to line 9030 in Listing 2. In this case, we are assured that the range of the number is 0 to 8. Therefore, we extract only a single char-

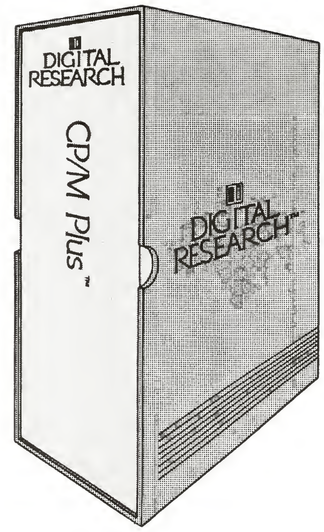
FREE CP/M-Plus™

from MAGNOLIA MICROSYSTEMS

New Reduced Prices

Prices effective 1/1/84

With purchase of 128K RAM Board at ~~$595~~ *Now $495*

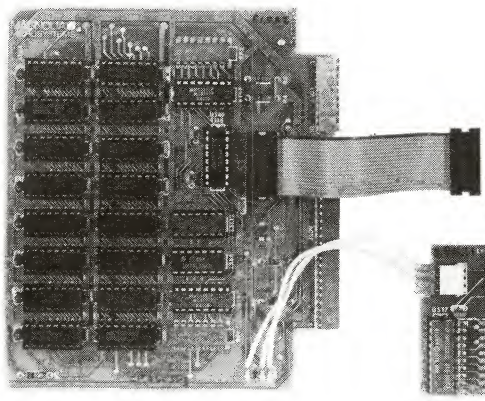


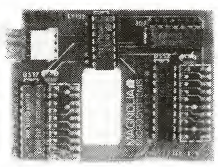
Digital Research's list price for CP/M-Plus is $350, but it will be included AT NO EXTRA CHARGE with purchase of our 77318 128K RAM board

CP/M-Plus™ support is available for Z89 and Z90 computers with our 128K RAM board and either the Z89-37 (Z90) or our own 77316 Double Density disk controller. Over 700K of files are included, so it will not be distributed on Hard Sectored media.

Using banked RAM, disk performance is enhanced through Hashed Directory tables, Directory Buffers, and LRU Data Buffers.

New utilities and features include a 'Help' command; optional Password protection and Time and Date Stamping of files; Console Redirection to or from disk files; and many others.





Our implementation of CP/M-Plus REQUIRES the use of our 128K RAM board. We have no plans to implement an un-banked memory version, most of the advantages of CP/M-Plus are neither available (nor practical) in an un-banked version.

Our BIOS supports both our 77316 and Zenith's Z89-37 Double Density controllers, Heath's H88-1 (Z17) Single Density controller, and our 77314 CORVUS and 77320 SASI-bus interfaces.

We are including the SOURCE code for our BIOS, together with Digital Research's MAC, RMAC, LINK, and SID software development tools, you may customize it to your specific application.

Orders must specify the Double Density controller and boot drive: the Z89-37 (as used in the Z90) or our 77316 (5-inch or 8-inch).

Earlier purchasers of the 128K RAM board, who are registered owners of Magnolia Microsystems CP/M 2 (S/N 2-175-xxxx) can update to CP/M-Plus for $100 (plus shipping and handling).

DISK INPUT/OUTPUT (I/O) BOARDS

These boards are made available apart from our subsystems for use by 'Systems Integrators'. End-users should carefully consider their skills and desired level of involvement before undertaking this responsibility; purchasing a complete subsystem from a qualified integrator may be a wise choice.

Although similar in name and apparent function, these boards fall into two distinct classes:

- INTERFACE boards connect a disk subsystem (containing a controller) to the computer. These boards include the Z89-47 and Z89-67 and our 77314, 77317, and 77320 interfaces.
- CONTROLLER boards actually control a plain disk drive. These boards include the H88-1 (Z17) and Z89-37, and our 77316 Double Density Controller.

All Magnolia I/O boards use proprietary techniques to expand the 89's I/O addresses, allowing use of more than Zenith's 5 (3 serial ports, 2 disk controllers). Since our INTERFACE boards also contain 3 RS232 ports (replacing the machine's H88-3) allowing expansion of the I/O capability without using an additional card slot.

Each Magnolia I/O board package includes:

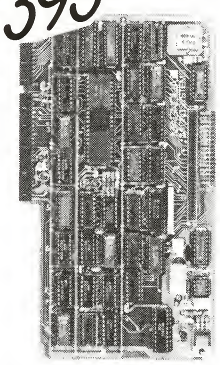
- The disk I/O board itself
- A new I/O decoder PROM
- A new Monitor EPROM
- A copy of MMS CP/M 2.2

Magnolia's CP/M software supports over 20 soft-sectored media formats, including all existing Zenith formats.

MAGNOLIA MICROSYSTEMS, INC. • 2264 - 15th Ave West
(206) 285-7266 (800) 426-2841 • Seattle, Washington 98119

Double Density Disk Controller
order number 77316

$395 ~~$495~~

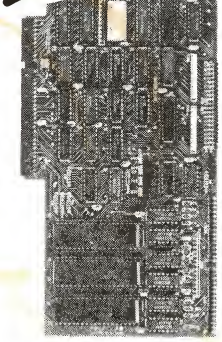


This board adds complete hardware and software support for four 8" single- or double-sided AND four 5" SS or DS, 48- or 96-track-per-inch (40- or 80- track) drives, at both single and double recording densities.

This controller is compatible with 8" drives by Shugart, Qume, and Tandon; 5" drives by Siemens, Tandon, and MPI; and most other 'industry standard' drives (possibly depending on the skill level of the experienced 'Systems Integrator').

SASI-bus Winchester Interface
order number 77320

$250 ~~$350~~



This board (functionally replacing the Z89-67 interface) contains 3 RS232 ports and can support up to 8 SASI-bus disk controllers, with up to 4 drives on each!

It is compatible with Winchester controlers by Xebec, DTC, Shugart, and others, as well as Zenith's Z67. CP/M software support for popular controller/drive combinations is included.

acter, without bothering to calculate the length of the string.

Line 10090 prints the single character string which contains the millions digit, and then it prints a comma.

Line 10100 sets Z2 equal to the hundred thousands, ten thousands, and thousands digits. Note that Z2 may be any value in the range 0 to 999. For numbers such as 6,000,123, Z2 will be 0. Since similar logic is required for printing these digits and the last three of the number, a nested subroutine (lines 10500-10560) is called.

Line 10120 is entered with the cursor positioned immediately after the thousands digit. If we are suppressing insignificant zeroes, the variable Z$ will contain a single space. If we have entered the range of the number, Z$ will contain the character "0". We accordingly print a space or a comma (at line 10130).

Line 10140: Since Z3 is the value of the millions digit and Z2 is the value of the hundred thousand, ten thousand, and thousands digits, this calculation will set Z4 to a value which is that portion of the number less than 1,000.

Line 10150 moves the value just determined to Z2, since the subroutine at 10500 works with Z2. Upon return from the nested subroutine at 10500, all the appropriate characters have been printed and the print head is guaranteed to be 10 positions to the right of its position at entry.

Line 10160 is the normal return to the routine which called line 10000.

Lines 10500-10560 are a nested subroutine used by the program to print groups of three digits with the possibility of suppressing leading zeroes.

Line 10500 checks to see if the value on entry is zero. If so, we must either print three spaces (if we are suppressing leading zeroes) or three zeroes. Since the value of Z$ is either a space or a zero character (as set by line 10070 or by line 10075), we can simply print Z$ three times and return.

Line 10510 is similar in appearance and function to line 9030 in Listing 2. In this instance, Z2$ will be a string that is 1, 2, or 3 characters long and represents the value of Z2.

Line 10520 tests if Z2 is 100 or larger. In this case, Z2$ is assured to be three characters with no leading zeroes. The string is printed.

Line 10530 is entered only if the number Z2 is 99 or smaller. We test to see if the value is larger than 9. If it is, then we are working with a number in the range 10, 11, ..., 99. We print the character in Z$ (a space or zero depending on whether zeroes are being suppressed or not) and then we print the two characters in Z2$.

Line 10540 is entered only if the value of Z2 is 1, 2, ..., 9. We print the space or zero in Z$ twice and then print the single character which is in Z2$.

Line 10550 is entered only when a non-zero has been passed to this subroutine. Now this may occur on printing of the digits for values above 1,000 or for values below 1,000 or both. Consider these possibilities in turn.

On printing of digits of values above 1,000: In this case, the next call to this nested subroutine will be to print digits for values less than 1,000. Since the number is larger than 1,000, we desire to not suppress leading zeroes on the next call.

On printing of digits of values below 1,000: In this case, the subroutine will not be called again until the larger subroutine is again invoked. The correct value of Z$ will be set by line 1070 or

On determining the accuracy of results

The following is provided for those readers desiring more insight into the limits of accuracy of the two subroutines.

B.H. BASIC stores all numbers internally as "floating point." Four bytes are allotted to each number: three bytes for the "mantissa" and one byte for the two's exponent. The "mantissa" is really just the binary representation of the number with an assumed "binary point" (as opposed to the decimal point) at a location indicated by the exponent. The examples in Table 1 should illustrate.

The exponents are all scaled by 200 octal (Q). An exponent of 177Q means −1. An exponent of 200Q means 0. An exponent of 201Q means +1. When the exponent is zero, the binary point implicitly belongs between the first and second bits of the mantissa. If more than zero, it moves the corresponding positions to the right; if less, to the left. (Remember: octal. So an exponent of 227Q will be +23.)

Consideration of the algorithm in Listing 3 is somewhat simpler than Listing 2, so we will address the accuracy of integer numbers first. In Table 2, observe the progressive use of more and more binary digits to represent decimal numbers as each power of two is approached and exceeded.

Observe that the orderly binary counting breaks down at 8,388,607. As mentioned in the body of the article, the algorithm fails after 8,388,607. To accurately handle higher numbers, we would need the extra bit indicated in parentheses.

This entire discussion presumes ideal, "normalized" representations of the numbers. That is, representations in which the maximum range of accuracy is available. In the course of calculations, it is possible to generate intermediate results which do not use the entire range. Therefore, the user is cautioned that it may be possible to obtain unexpected results.

Consider, by the way, that the Benton Harbor BASIC manual flatly states that the largest integer number is 65,535. This, of course, does not seem to be the case at all. I am not sure why the manual makes this statement. B.H. BASIC, however, does not seem to distinguish between integer types and floating point, etc., except when specifying arguments for functions. (PEEK, for instance, takes an integer argument.) In such a case, the limitation is entirely reasonable.

Fractions

Consider now the representation of decimal fractions in floating point notation. Representing whole numbers, of course, is no problem. But while we can easily represent some decimal fractions (.5 and .25, for instance), tenths and hundredths turn out to be irrational numbers in binary. (.1 equals .0001100110011...

Decimal Number	Binary Number	Binary Mantissa	Exponent
1.0	1.0	01000000 00000000 00000000	201Q
0.5	0.1	01000000 00000000 00000000	200Q
0.25	0.01	01000000 00000000 00000000	177Q
2.0	10.0	01000000 00000000 00000000	202Q
3.0	11.0	01100000 00000000 00000000	202Q
4.0	100.0	01000000 00000000 00000000	203Q

Table 1: Examples of how Benton Harbor BASIC stores numbers. Four bytes are allotted to each number: three bytes for the "mantissa," one for the "exponent." The mantissa is the binary representation of the number with an assumed "binary point" indicated by the exponent. An exponent of 200Q would place that point between the first and second bits of the mantissa.

1075. No harm has been caused.

In summary, although Benton Harbor BASIC does not have double-precision arithmetic and it does not support PRINT USING, this is no hindrance to preparation of neatly aligned columnar data. With due consideration to the scale of values to be represented, meaningful and attractive displays can be prepared.

References

Software Reference Manual, Cassette System, Extended Benton Harbor BASIC, Heath Part #595–2347, Heath Company, 1979.

Software Reference Manual, HDOS System, Extended Benton Harbor BASIC, Heath Part #595–2479, Heath Company, 1980.

Decimal Number:	Binary Mantissa	Exponent
32,766	01111111 11111110 00000000	217Q
32,767	01111111 11111111 00000000	217Q
32,768	01000000 00000000 00000000	220Q
32,769	01000000 00000000 10000000	220Q
4,194,304	01000000 00000000 00000000	227Q
4,194,305	01000000 00000000 00000001	227Q
8,388,606	01111111 11111111 11111110	227Q
8,388,607	01111111 11111111 11111111	227Q
8,388,608	01000000 00000000 00000000(0)	230Q
8,388,609	01000000 00000000 00000000(1)	230Q
8,388,610	01000000 00000000 00000001(0)	230Q

Table 2: Examples to illustrate the progressive use of more binary digits to represent decimal numbers as each power of two is approached and exceeded. Observe that the orderly binary counting breaks down at 8,388,607. As mentioned in the body of the article, the algorithm fails after 8,388,607. To accurately handle higher numbers, we would need an extra bit—indicated by the digits in parentheses.

ad infinitum.) So we have to make do with approximations. Table 3 shows a sample to produce the binary equivalent of .99.

It can be seen that at least eight binary digits are required to represent decimal values in the range sought by the algorithm. If eight are taken from the total of 23 bits available for the entire mantissa, then the largest integer part of the number would of necessity be limited to 15 bits. Now,

$2^{15}-1=32,767$

On test, the program is observed to fail at values exceeding 32,767.90.

Testing of the algorithm in Listing 2 was somewhat ambiguous. Use of FOR...NEXT loops with the STEP increment set to .01 yielded inaccurate results on several occasions. In each case, the value of Z was then set to the indicated failing value and the subroutine was again called. In each case, the subroutine returned the correct print field. The conclusion is that for some values, the B.H. BASIC implementation of FOR...NEXT can yield results which vary by as much as .001 decimal if the increment is .01.

Additionally, the program in Listing 2 adds .005 to each value it receives in order to round the value to the nearest one hundredth. However, .005 is only imperfectly represented with eight binary digits. Binary .00000001 or decimal .00390625 is in fact nearly 22% off the precise value.

So the user is reminded that the accuracy of the intermediate results may influence the accuracy of the values passed to the routine.

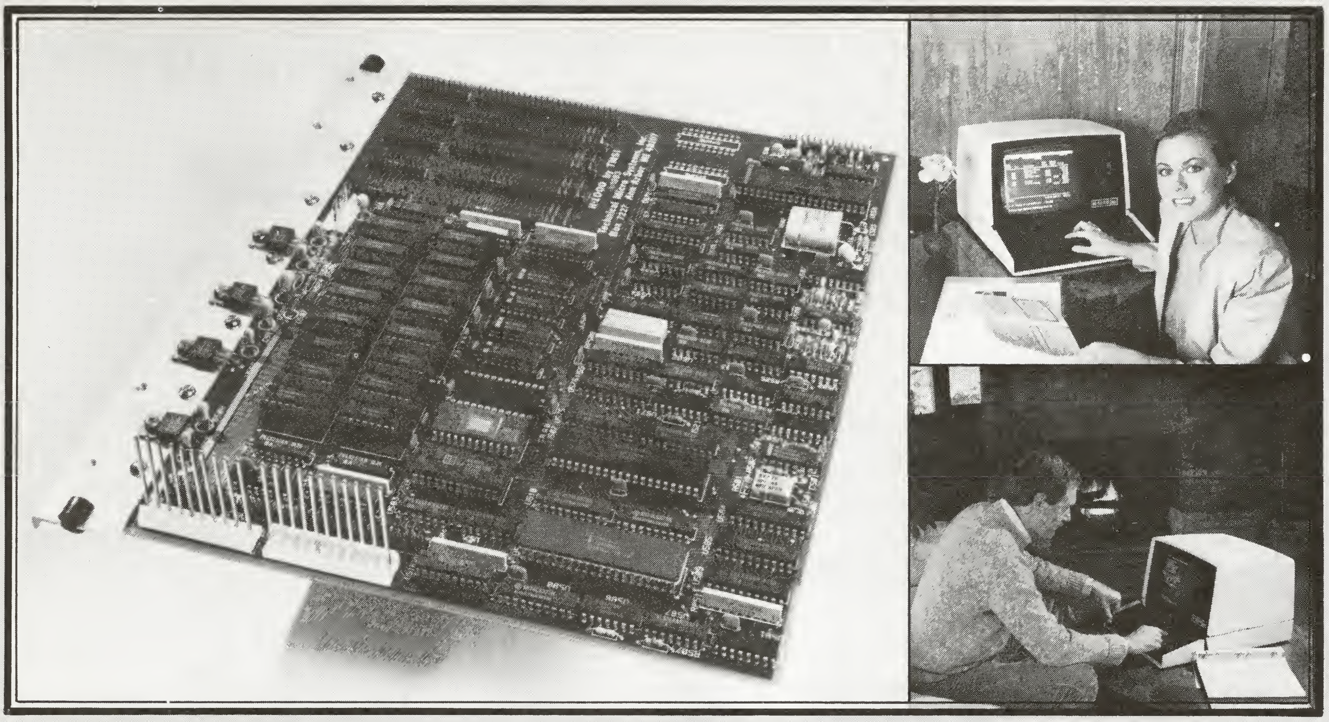
Binary Number:	Decimal Value:
.1	.5
.01	.25
.001	.125
.0001	.0625
.00001	.03125
.000001	.015625
.0000001	.0078125
.00000001	.00390625
.11111111	.99609375

Table 3: A sample to produce the binary equivalent of .99.

References

Software Reference Manual, Cassette System, Extended Benton Harbor BASIC, Heath Part #595–2347, Heath Company, 1979.

Software Reference Manual, HDOS System, Extended Benton Harbor BASIC, Heath Part #595–2479, Heath Company, 1980.



You can be a performance leader in business or at home with the H-1000.

Great News for Heath & Zenith Owners!

A Z80/8086 Upgrade for the H89/Z89

The H-1000 is a plug in replacement for your present CPU board, giving you twice the speed, twice the I/O capacity and four times the memory. Yet it's totally compatible with all your present H89/Z89 hardware and software. Best of all, we've added the 8086, a true 16-bit CPU for the high performance software of the 80's. You get compatibility with such machines as Zenith Z-100 and IBM PC at a fraction of the cost of a new system.

Regardless of the operating system, you get virtually all the benefits of an additional disk drive, even if you only have one. Our "ghost disk" system loads your disk into the memory. This becomes the extra disk drive and leaves your real drive free for another disk. RAM is many thousands of times faster than the best disk drive, so you get practically instantaneous access to information. That's important for programs requiring frequent access to large data bases, or if you simply don't like waiting while your computer talks to your disk drive.

TMSI has taken the best features of your 8-bit computer and combined them with the best of the 16's to give you a package that's state-of the-art... and the ONLY package that lets you build on all your current investment...instead of starting over!

The H-1000 comes safely packaged, fully assembled and tested. It can be installed in under an hour using the manual provided. No soldering or special tools are required. A full 90 day warranty covers each unit with an optional service contract available.

Compare this performance leader to other systems. We think TMSI gives you your best choice.

COMPATIBILITY CHART	8/16 BIT SYSTEM	16 BIT SYSTEM	8 BIT SYSTEMS		8/16 BIT SYSTEM
	Z100	IBM PC	H89/Z89	DG SUPER 89	H-1000
Runs Z80 and 8080 code	no	no	yes	yes	yes
HDOS 2.0 compatible	no	no	yes	yes	yes
CP/M compatible	yes	no	yes	yes	yes
Runs all H89/Z89 software	no	no	yes	yes	yes
Supports all H89/Z89 hardware	no	no	yes	yes	yes
8-bit clock	5MHz	none	2MHz	4MHz	2/4MHz
16-bit clock	5MHz	4.7MHz	none	none	8MHz
Maximum memory	756k	756k	64k	256k	1048k
Runs MS—DOS and CP/M-86	yes	yes	no	no	yes
color graphics	yes	yes	no	no	no
price/performance	ouch	ouch	good	better	best

Complete systems as well as the H-1000 board are available. For more information and dealer listings, call or write NOW. Dealer inquiries invited.

TMSI

TMSI
Technical Micro Systems, Inc.
P.O. Box 7227, Dept. A
366 Cloverdale
Ann Arbor, MI 48107
(313) 994-0784

Circle #173 on Reader Service Card

Create Forms on Your MX-80

You can use a dot-matrix printer to make high-quality forms.

Leonard E. Geisler

Let's face it: notes or data written on an official-looking form have more attention-getting power than notes hastily scribbled on the backs of envelopes or laundry lists! Come to think of it, that old laundry list is a *form* which allows a clerk to gather all the necessary data from customers so that their laundry is done expeditiously and to their satisfaction.

And the same applies to a shopping list, to your weight-loss program, the children's daily chores-and-rewards lists, community billboard information request posting forms, custom-designed "certificates," short-wave listener's (SWL) logs, gas mileage, tolls and up-keep forms for all your car pool members, and so-endlessly-on!

So I am presenting here a practical "Form Formula" which should help you create and print your own private, limited-edition, standardized forms. This is a "how-to" for those readers who may occasionally have to create business or other forms with their H8/89 computer and Epson MX-80 printers. In this article, I'll try to show you the complete steps to design and print the heading and body of a form.

The Heath Users' Group (HUG) has published numerous and helpful articles and notes in its journal, *REMark*, from which I've garnered considerable knowledge and material for this article. I've also flipped the pages of my Epson manual until they're worn out, preparing these form examples.

Caveat

Before you try any of the ideas I've put together here, you must have Heath's LPMX80.DVD printer input/output (I/O) program. It comes with the Heath Disk Operating System (HDOS) version 2.0 and you should install it on your Benton Harbor BASIC working disk. Rename this driver as LP.DVD. Make sure the internal switches of your MX-80 are set as I've indicated in Table 1; and that the appropriate computer serial I/O port is connected properly to Heath's interface card in the MX-80. The interface switches are set for 4800 baud, LPMX80.DVD's "default" value.

Boot your working BASIC disk, then type: SET LP: HELP (don't forget the carriage return <CR>) and study the SET option menu. Now type these settings into LP.DVD:

NODOUBLE	allows program density control
NOEMPH	allows program density control
WIDTH 132	allows 10 or 17 characters/inch print
FORM	feeds next sheet of paper
PAGE 60	number of typed lines on sheet

Don't fool with any of the other LP.DVD (LPMX80.DVD) SET options, except to turn them off if you've set them other than as PIPped from the distribution disk. This will keep you from experiencing a lot of strange results as your program fights LP.DVD.

The examples presented here employ MX-80 graphics and other important printer commands to expand, emphasize, and otherwise make the finished work reasonably attractive, yet quite useful. Use these ideas as a basis for your own designs.

Step 1—Design a heading

This will use a lot of printer paper, as it is very difficult to space MX-80 graphic characters and double-wide, double-dark letters to fit nicely on your paper. (That neat example of the MX-80's capabilities on page 65 of Epson's manual probably took several eight-hour days to program.) I recommend you dig out all that scrapped printout you've squirreled away and use its blank side until you're satisfied your program is what you want.

a. Print an alignment rule on the top and bottom of several sheets of blank paper. Use the program in Listing 1 by itself and rewind the paper for over-printing, or combine Listing 1 with your preliminary header as in my example.

b. Write a short "printer" program which tabs all the lines in your heading the same distance from the left margin and run it on one sheet of paper. For starters tab just three spaces.

c. Check the printout for spelling and punctuation and correct errors before continuing. Re-do your "printer" as needed and run another—correct—copy.

d. Lay the printout on a flat surface and use a ruler to draw connecting (colored) pencil lines between similar points on the top and bottom printed ruler scales.

e. Count the spaces between the last character of each line and the right-hand margin to determine which way you have to shift each line to make it symmetrically centered on the sheet. If you've used double-width characters, this may take several tries before it comes out. (Courage & Fortitude!!)

f. When satisfied your heading is just right, delete the ruler-printer lines and run a "proof copy" to verify its appearance. Use up all that scrap computer printout—paper's expensive.

Step 2—Design the body

Study the complete program listing to see how I constructed it. You might try putting in a GOSUB as I did in Listing 1 to print a scale at the top and bottom of your trial printouts. Note the GOSUB at line 270; it eliminates a lot of program typing and your MX-80 will buzz right along. Notice that there are 16 loops to print eight lines of dashes and eight double-spaces. Adjust the subprogram to fit your requirements. Also note that it is easier to type in the correct number of dashes when combined with words than to try to use another GOSUB. Also, note the way I set up my program so it asks

MX-80 Internal Switches				8141 Serial Interface	
Switch	Setting	Switch	Setting	Switch	Setting
1–8	ON	2–4	OFF	1–1	ON
1–7	OFF	2–3	OFF	1–2	OFF
1–6	ON	2–2	ON	1–3	ON
1–5	ON	2–1	ON	1–4	OFF
1–4	OFF			1–5	ON
1–3	OFF			1–6	OFF
1–2	ON			1–7	OFF
1–1	ON			1–8	n/a

Table 1: These are the appropriate settings for the MX-80 for use with the form-printing procedures described in the article.

you for and prints just the number of copies you want. Of course, I could have used Benton Harbor BASIC's CIN(0) function, but then you wouldn't have any fun, programming that in!

I'm not going to go into details here, since your form will be different from my example. Use it as a guide while composing your own.

Step 3—Fancy stuff

I have never seen a form that did not have "fine print" on it somewhere! Back in the paragraph on setting your LP.DVD, I told you to set WIDTH 132. Here's where you'll need it. And you'll need a double set of form spacing rulers to help in composing those lines of fine print so they look professional. See Listing 2 for a program which prints both alignment rulers. In this listing, I've included text from my example to give you an idea of how to proceed. Be sure to include blank spaces in text lines where they end with a double quote and semicolon (";). If you don't, you'll have words all run together. (See my example.) Of course, this can be used to split words you want to print as a whole, if they don't fit into one program line.

Note that I held all text lines here to about 56 spaces between BASIC quotes. This helps prevent "wraparound" and makes it much easier to read your program hardcopy. (You *do* make one for backup, don't you!?) It also makes it easier to compose text to fit into those 132 spaces (56 + 56 + 20 = 132).

And then there's *mixed-size* printing. See how I've combined Epson's CHR$(15) (narrow letters) with CHR$(14) (wide letters) modes? I think this is an elegant

```
00005 REM   FORMRULE  VER 1.1  30 JAN 83
00010 REM   by L.E. Geisler, Ann Arbor, Michigan 48105
00015 REM     May be reproduced only if author is given credit.
00020 REM   SET LP: NOFORM and use printer LF to advance paper & view printout
00025 REM   Delete all REM lines from finished program to conserve memory
00030 POKE 12121,255
00035 REM   Allows access to MX-80 graphics by BASIC
00040 OPEN "LP:"FOR WRITE AS FILE #1
00045 REM   Fetch form rule printer sub program
00050 GOSUB 5000
00055 REM   One rule printed, continue program
00060 PRINT #1,CHR$(27)"E"CHR$(27)"G"
00065 REM   Tells printer to print double-strike, emphasized characters
00070 PRINT #1,CHR$(14)"PRECISION COMPUTER EQUIPMENT SERVICE CO."
00075 REM   tells printer to print double-width characters, print 1st line
00080 PRINT #1,
00085 REM   feeds one blank line
00090 PRINT #1,TAB(5)"A Healthy Computer Pays For Itself!"
00095 REM   prints slogan in double-strike, emphasised mode
00100 PRINT #1,CHR$(27)"F"CHR$(27)"H"
00105 REM   Tells printer to go back to ordinary single-strike letters
00110 GOSUB 5000
00115 REM   Print another alignment rule
00120 CLOSE #1:END
00125 REM   All done.
00130 GOSUB 500
00140 CLOSE #1:END
00145 REM   BE SURE TO RE-SET LP: FORM before printing out finished work!
05000 REM   Here's a ruler for you to use for judging copy alignment
05010 FOR I=1 TO 8:PRINT #1,"|.... ....";:NEXT I:PRINT #1,:RETURN
```

Listing 1: FORMRULE.BAS; this program may be used to print an alignment rule. This incorporates the heading of the example form shown in Figure 1.

```
|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....

PRECISION  COMPUTER  EQUIPMENT  SERVICE  CO.

    A Healthy Computer Pays For Itself!

|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....


LINE 90 TAB CHANGED FROM 5 TO 24 FOR 2ND RUN;  ITS CENTERED!


|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....

PRECISION  COMPUTER  EQUIPMENT  SERVICE  CO.

                       A Healthy Computer Pays For Itself!

|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....
```

Figure 1: The output of Listing 1, with an alignment rule. (Reduced 25% in size.)

```
00010 REM   FINEPRNT  VER 1.5  29 JAN 83
00020 REM    By L.E.Geisler, Ann Arbor, Michigan 48105
00025 REM    May be reproduced only if author is given credit.
00030 OPEN "LP:"FOR WRITE AS FILE #1
00035 GOSUB 2000
00040 PRINT #1,CHR$(15)
00045 GOSUB 3000
00047 PRINT #1,
00050 PRINT #1,CHR$(14)"CUSTOMER PLEASE NOTE: "CHR$(20)"This report is used ";
00060 PRINT #1,"for billing purposes.   If service call exceeds contracted ";
00065 PRINT #1," time, it"
00070 PRINT #1,"becomes a one-time-only addendum to your original contract ";
00075 PRINT #1,"with us for the period covered by dates and times entered a";
00080 PRINT #1,"bove by our";
00085 PRINT #1,"Service Technician.  Your invoice will then show original ";
00090 PRINT #1,"contracted charges, plus subsequent accrued charges."
00100 PRINT #1,
00110 PRINT #1,CHR$(14)"SERVICE TECHNICIAN:  "CHR$(20)"Fill out form complete";
00120 PRINT #1,"ly, date and sign it.  Have customer's representative read ";
00125 PRINT #1,"and"
00130 PRINT #1,"sign it.  Give original to customer.  Return carbon copy an";
00135 PRINT #1,"d your expense report to our billing department.  List all ";
00140 PRINT #1,"time"
00145 PRINT #1,"worked, including starting and finishing times.  ";
00180 PRINT #1,CHR$(14)"BE ACCURATE."CHR$(20)
00185 GOSUB 3000
00190 PRINT #1,
00205 PRINT #1,CHR$(18):GOSUB 2000
00210 CLOSE #1:END
01999 REM   HERE'S A 10-CPI FORM RULER
02000 FOR I=1 TO 8:PRINT #1,"|.... ....";:NEXT I:PRINT #1,:RETURN
02999 REM   HERE'S A COMPRESSED-MODE FORM RULER; NOTE DIFFERENT SPACING
03000 FOR I=1 TO 14:PRINT #1,"|.... ....";:NEXT I:PRINT #1,:RETURN
03010 REM   DON'T WORRY ABOUT LAST PRINTED UNIT WRAPPING TO NEXT LINE.
```

Listing 2: FINEPRNT.BAS**; this program prints a double set of form spacing rules to help in composing lines of fine print.**

```
|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....

|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|.
...^....

CUSTOMER PLEASE NOTE: This report is used for billing purposes.   If service call exceeds contracted  time, it
becomes a one-time-only addendum to your original contract with us for the period covered by dates and times entered above by ourSer
vice Technician.  Your invoice will then show original contracted charges, plus subsequent accrued charges.

SERVICE TECHNICIAN:   Fill out form completely, date and sign it.  Have customer's representative read and
sign it.  Give original to customer.  Return carbon copy and your expense report to our billing department.  List all time
worked, including starting and finishing times.  BE ACCURATE.
|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|.
...^....


|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....
```

Figure 2a: The initial output of Listing 2, with a double set of form spacing rules. Note the inappropriate line endings, which must be repaired. (Reduced 35% in size.)

```
|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....

|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|.
...^....

CUSTOMER PLEASE NOTE: This report is used for billing purposes.   If service call exceeds contracted  time, it
becomes a one-time-only addendum to your original contract with us for the period covered by dates and times entered above  by  our
Service Technician.  Your invoice will then show original contracted charges, plus subsequent accrued charges.

SERVICE TECHNICIAN:   Fill out form completely, date and sign it.  Have customer's representative read and sign
it.  Give original to customer.  Return carbon copy and your expense report to our billing department.  List all time worked,
include all starting and finishing times.  BE ACCURATE.
|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....|.
...^....


|....^....|....^....|....^....|....^....|....^....|....^....|....^....|....^....
```

Figure 2b: The output of Listing 2 (with line endings repaired), ready to be incorporated into the main listing. (Reduced 35% in size.)

touch which emphasizes the fine-print paragraphs enough so they'll be read. I admit it does call for a little extra alignment work, but it's worth it. You might even use it to tack footnote references on the bottoms of those scholarly reports, even though it means you have to run the word processor first and BASIC second to print them! (I have to—my text processor doesn't speak Epson!)

Conclusion (at last!)

I've tried to present enough practical, step-by-step instructions and hints to allow the computer novice to turn out reasonably acceptable business-type forms (for fun or profit, or both). With these instructions and accompanying examples, you should have little or no trouble in doing a very professional-looking job. And maybe your wife—if she's like mine—may become able to see your computer as something more than an expensive toy.

Listing 3: REPFORM.BAS; the program to print the entire example form.

```
00005 REM   REPFORM -- REPORT FORM PRINTER PROGRAM  VER 1.2  30 JAN 83
00010 REM   by L.E. Geisler, Ann Arbor, Michigan 48105
00015 REM   May be reproduced only by permission of author.
00017 CLEAR
00030 POKE 12121,255
00035 REM   Allows BASIC to access MX-80 graphics escape codes
00040 INPUT "HOW MANY SERVICE REPORTS?  ";N
00050 FOR I=1 TO N
00055 REM   Here's a counter routine
00065 REM   It stops output when specified number of forms have been printed
00070 GOSUB 85
00075 NEXT I:REM   Note use of repetitive GOSUBs.
00080 END
00085 OPEN "LP:"FOR WRITE AS FILE #1
00090 REM   Let's get fancy & box our heading!
00095 PRINT #1,CHR$(27)"E"CHR$(167);
00100 FOR N=1 TO 78:PRINT #1,CHR$(163);:NEXT N:PRINT #1,;CHR$(203)
00105 REM   Top bar & corners done, now do the rest
00110 PRINT #1,CHR$(14)"PRECISION COMPUTER EQUIPMENT SERVICE CO."CHR$(20)
00117 PRINT #1,CHR$(181);
00118 PRINT #1,TAB(23)"A Healthy Computer Pays For Itself!";
00120 PRINT #1,SPC(22)CHR$(202)
00130 PRINT #1,CHR$(173);
00135 FOR N=1 TO 78:PRINT #1,CHR$(172);:NEXT N:PRINT #1,;CHR$(174)
00140 PRINT #1,CHR$(27)"F":REM  That's done, now let's print a service form!
00145 PRINT #1,:REM   Call up emphasized printing for easy reading
00150 PRINT #1,CHR$(27)"G"TAB(26)">> S E R V I C E  R E P O R T <<"
00155 PRINT #1,:REM   Pay close attention wherever semicolons (;) are used!
00160 PRINT #1,"Customer______________________________";
00165 PRINT #1,SPC(4)"Complaint_____________________________"
00167 PRINT #1,
00170 PRINT #1,"Address_______________________________";
00175 PRINT #1,SPC(4)"______________________________________"
00180 PRINT #1,:REM   38 characters/block + 4 spaces = 80 columns, total
00185 PRINT #1,"______________________________________";
00190 PRINT #1,SPC(4)"P/O Number____________________________"
00195 PRINT #1,
00200 PRINT #1,"Phone____________________Xtn_________";
00205 PRINT #1,SPC(4)"Remarks_______________________________"
00210 PRINT #1,
00215 PRINT #1,"Contact_______________________________";
00220 PRINT #1,SPC(4)"______________________________________"
00225 PRINT #1,
00230 PRINT #1,"Service Tech. Check in/out time/date____";
00235 PRINT #1,"________________________________________"
00240 PRINT #1,
00245 PRINT #1,"Work done_______________________________";
00250 PRINT #1,"________________________________________"
00255 PRINT #1,
00260 FOR J=1 TO 16
00265 FOR J=1 TO 16
00270 LET J=J+1:GOSUB 2000
00275 NEXT J
00280 PRINT #1,
00281 PRINT #1,"Svc Tech sign_________________________";
00282 PRINT #1,SPC(4)"Customer______________________________"
```

```
00283 PRINT #1,
00285 PRINT #1,CHR$(27)"H":REM   Turns off emphasized printing mode.
00290 PRINT #1,CHR$(15):REM   Here's where we do the FINE print!
00300 PRINT #1,CHR$(14)"CUSTOMER PLEASE NOTE:  "CHR$(20)"This report is us";
00310 PRINT #1,"ed for billing purposes.  If service call exceeds contracted"
00320 PRINT #1,"time, form becomes a one-time-only addendum to your original";
00330 PRINT #1," contract with us for the period covered by dates and times";
00340 PRINT #1," entered"
00350 PRINT #1,"above by our Service Technician.  Your invoice will then show";
00360 PRINT #1," original contracted charges, plus subsequent accrued charges."
00370 PRINT #1,
00380 PRINT #1,CHR$(14)"SERVICE TECHNICIAN:  "CHR$(20)"Fill out form complete";
00390 PRINT #1,"ly, date and sign it.  Have customer's representative read and"
00400 PRINT #1,"sign it.  Give original to customer.  Return carbon copy and ";
00410 PRINT #1,"your expense report to our billing department.  List all time"
00420 PRINT #1,"worked, including starting and finishing times. "CHR$(14)"BE ";
00430 PRINT #1,"ACCURATE."
00440 PRINT #1,CHR$(27)"H":CLOSE #1:RETURN
02000 FOR Z=1 TO 80:PRINT #1,"_";:NEXT Z:PRINT #1,:PRINT #1,:RETURN
```

PRECISION COMPUTER EQUIPMENT SERVICE CO.

A Healthy Computer Pays For Itself!

>> S E R V I C E R E P O R T <<

Customer________________________ Complaint________________________

Address________________________ ________________________

________________________ P/O Number________________________

Phone________________Xtn________ Remarks________________________

Contact________________________ ________________________

Service Tech. Check in/out time/date________________________

Work done____

____

____

____

____

____

____

____

____

Svc Tech sign________________________ Customer________________________

CUSTOMER PLEASE NOTE: This report is used for billing purposes. If service call exceeds contracted time, form becomes a one-time-only addendum to your original contract with us for the period covered by dates and times entered above by our Service Technician. Your invoice will then show original contracted charges, plus subsequent accrued charges.

SERVICE TECHNICIAN: Fill out form completely, date and sign it. Have customer's representative read and sign it. Give original to customer. Return carbon copy and your expense report to our billing department. List all time worked, including starting and finishing times. **BE ACCURATE.**

Figure 3: This is the output of Listing 3, showing a sample form. You can modify the program to produce your own forms. (Reduced 25% in size.)

Circle #145 on Reader Service Card

CHUGCON 83: Sharing Expertise

The world's largest local Heath Users' Group brought together users from the region and vendors from around the country.

Frederick Zimmerman

The Capital Heath Users' Group held its second annual conference, CHUGCON 83, on Saturday, October 22. It took place at the Tyson's Westpark Hotel in suburban Tyson's Corner, Virginia, just outside Washington, D.C. The conference ran the whole day, so I got to attend a full program of high-quality conference activities. I thought CHUG did a thoroughly professional job of organizing its regional users' group conference.

Over 600 people came to CHUGCON 83. More than 30 vendors showed Heath/Zenith hardware and software products in a crowded exhibition hall. (CHUG officials had to turn down at least a dozen vendors who wanted to exhibit at the last moment.) Technically knowledgeable CHUG members shared their expertise in seminars on subjects such as Z100 hardware and software, dBASE II, and the programming language C.

CHUG also arranged for speakers at lunch and dinner. Dr. Ruth Davis, a respected figure in the world of mainframe computing, gave a luncheon speech in which she shared her opinions about the role of microcomputer users' groups in social change. The first speaker at the CHUG dinner banquet, HUG software developer Terry Jensen, offered some well received inspirational remarks. And the second speaker at dinner, HUG manager Bob Ellerton, announced that the Heath Company is giving members of the national HUG a discount on all Heath/Zenith computer products.

The exhibition area

I thought the most noteworthy aspect of the exhibition area was that it was about as large as the vendor exhibit at the first National Heath Users' Group Conference, HUGCON 1, held in Chicago in August 1982. The exhibition room itself was, if anything, slightly nicer than the room for HUGCON 1. There were fewer attendees at CHUGCON 83, but about the same number of vendors—many of whom travelled a considerable distance to exhibit.

For example, Trionyx Electronics came from California; Hoyle & Hoyle Software and GROFFics came from

Photo by Elizabeth Saxon

Photo by Elizabeth Saxon

The theme of CHUGCON 83 was "educational and scientific interchange". In the photograph at left, Steven G. Meyerson (left) discusses his *dBASE II Programmer's Notebook*, a 46-page "collection of techniques and hints . . . used in writing programs in dBASE II." ($12.95, P.O. Box 2027, Poquoson, VA 23662.) In the photograph at right, Bob Shon (center) and another CHUGCON attendee examine a sheaf of technical literature picked up from CHUGCON vendors. Larry Sites (left), who built his first Heathkit in 1949, evidently has some comments to offer.

Tutorials and demonstrations at CHUGCON

Speaker	*Topic*
Bob Shon	Computer Video Generation
Carl Fosler	Low Cost Data Base Management System Software
Ray Cherry	Introduction to C
Don Lewis	Public Domain Software—for the new user
Arthur Goldsmith	dBASE II Beginner
Robert Todd	Public Domain Software—for the experienced user
Mike Supley, Chair	Open Forum: Future Directions for CHUG
Bill Johnson	Genealogy and Computers
John Stetson	Z100 Software
Paul Beck	dBASE II Advanced
Mike Cogswell	Z100 Hardware
Leon Wittwer	Improving the H89
Terry Jensen	HUG Presentation

North Carolina; Technical Micro Systems, Inc., came from Ann Arbor, Michigan; Husker Systems came from Nebraska; New Orleans General Data Services came from Louisiana; and Extended Technology Systems from Pennsylvania.

Not everyone had to travel so far to exhibit. Many CHUG members were exhibiting products, and several local computer distributors had booths.

GROFFics had my favorite booth decoration, a Space Shuttle balloon hanging from the ceiling. Presumably, it was there to draw attention to the debut of Shuttle Lander, their flight simulator program which runs under the Heath Disk Operating System (HDOS).

Learning from CHUG's experts

CHUGCON attendees had a choice of two different seminars every hour from 8:30 to 5:00. These tutorials and demonstrations covered a wide range of hardware and software subjects. (A list of topics and speakers accompanies this article.) Of the ones I attended, I thought Ray Cherry's "Introduction to C" was the most entertaining. I liked his capsule description of C: it's like a powerful Kawasaki 1200 motorcycle—"with no brakes."

Luncheon: Dr. Ruth Davis

Dr. Ruth Davis has been director of the National Bureau of Standards' Institute for Computer Sciences and Technology; Deputy Under Secretary of Defense for Research and Advanced Technology; and Assistant Secretary of Energy for Resource Applications. She's now president of her own company, The Pymatuning Group, Inc., which specializes in science and technology management and development strategies. In short, an impressive speaker for a regional users' group conference!

The gist of her occasionally rather rambling remarks was that 1) microcomputers are changing the nature of our society; 2) economists, historians, and other social scientists don't understand those changes at all; and 3) computer hobbyists will be the first to know about such social changes. Computer hobbyists, Davis concluded,

Circle #240 on Reader Service Card

"are the kernel, the core, of a kind of world that nobody has described."

HUG to CHUG: Do good

HUG software developer Terry Jensen is also a SYStem OPerator on the Heath Users' Group Special Interest Group bulletin board on CompuServe's MicroNet. The HUG bulletin board plays a very important role in maintaining a sense of community among Heath/Zenith users. So it's particularly noteworthy that the most important message of Jensen's dinner speech was that Heath/Zenith users should not allow their community to become an exclusive clique for computer hobbyists only. Rather, Jensen argued, users' groups have a special responsibility to inform the general public about computers. His message was well received by the CHUG audience, which gave him the strongest applause of the day. (Excerpts from Terry's remarks accompany this article.)

HUG discount

The next speaker, National Heath Users' Group Manager Bob Ellerton, had welcome news for members of the national HUG: the Heath Company is now giving HUGgies a discount on Heath/Zenith products. Anyone who has been a member of the national HUG for 90 days or more is entitled to a 10% discount on all computer-related Heathkit products, and a 20% discount on all Heath Company wired computer products purchased through the local Heathkit Electronic Center or through mail order from Heath.

Ellerton said the HUG discount is a bottom-line discount, which means that it takes effect *after* any other discounts are applied. He supplied further details: the maximum purchase per year to which the discount may be applied is $5,000, and the HUG member may not apply the discount to more than one unit of each item purchased. ("You can't go out and buy three Z100s or anything like that.") HUG members will be issued new ID cards over the course of the next year, and equipment purchases will be recorded on the back of the new cards. Finally, the discount was to become effective as of December 1, 1983.

Ellerton said he believed that the discount for HUGgies "was brought about by the last two years of outstanding response to the national Heath Users'

Reach Out and Teach Someone

Terry Jensen

I would like to be serious for just a few moments and address two issues which I feel are important to each of us. Both of these issues relate to communication. At the conclusion of discussing these issues, I also would like to present a charge to you as a group and to you as individuals.

How many of you this evening have owned a computer for over, let's say, one year?

How many have owned a computer for over two years?

How many have been hackers for over three years?

I wonder how many of you have taken a microcomputer-related class to help you learn how to use your computer? Not a computer class but a *micro*computer-related class?

I believe we would find if we took a poll that the longer you have owned your computer, the less likely it is that you have taken a *micro*computer-related class. Most of the "hackers" have learned by jumping in there and trudging along.

How many of you hackers remember what it was like the first few months you had your computer? It was fun! Frustrating! Exciting! It was a mixture of many emotions. One session you added on that new board, the next session a new disk drive, and the next session you were boosting the power supply. One minute you were typing in a program, the next minute you were debugging the program.

For the past several months, I have had the opportunity of teaching microcomputer-related classes in the evening hours. Sixty-five percent to 70 percent of the students are over 40, with many of them presently retired. Sixty-five to 70 percent are women. (You may be interested in knowing that 10 to 15 percent of the students are teachers.)

What do you think are the two most popular responses to the question, "Why are you taking this class?"

"I want to know what these computers are all about."

"My kids or my grandkids are using the computer and I want to have an idea what is going on."

The large majority of people today do *not* understand what is "going on" with computers. They are afraid of computers. Most of these people are going to remain illiterate (computer speaking), because they are either not in a position to learn, or they don't want to learn, or they do not know how to learn what a computer is capable of, as the true hacker has learned. This new influx of unlearned and unprepared computer users has and will continue to change the way that we, the "experienced" users, confront or communicate with new computer users.

The person purchasing a computer today, generally speaking, expects the computer to do wonders at the press of a key. We, as computer hackers, know this is not the case. Many hours, days, months, and years go into the design and implementation of computer hardware and the software that produces an accurate and pleasing result. Miracles do *not* happen overnight. It is not just the press of a key that produces; it is only long hard work by engineers, designers, software writers, or what have you, that "produces."

But the computer buyer of today does not know that! He expects the computer to produce and to do it his way and it must be done now!

Is he missing the boat? Is he missing the whole picture? Sure he is! But why should he know better? How can he know the whole picture?

I guess that comes down to us! Those of us here in this room—those of us who have had a computer for more than one year—it is our responsibility to tell him! As we come in contact with this person maybe only one time or maybe day after day, we must remember his vantage point, we must be able to relate to him. Then we can begin to teach or communicate to him how to understand his computer.

But how can we do this?

Each time we are in contact with a new computer user, we must remember what it was like when we first began hacking on our computer. We must understand what he sees. We must begin to explain and educate (if you will) this person "bit by bit." Each time we help this user, he will come one step closer to understanding the "whole picture."

On the other side of the coin, we have the second issue. What do we do with the likes of the "414s"? (As you recall, they are the group of boys and

Group Conference and conferences like CHUGCON." Ellerton said that Bill Johnson, the president of the Heath Company, sent a memo to Joe Schulte—president of Veritechnology, the corporation which runs the Heathkit Electronic Centers—and Dave Altwies, Heath's director of marketing. Johnson told them: "Figure out how to do this." So, Ellerton said, "We sat down and argued it out." Perhaps with an eye towards pleasing his audience, he added: "Of course, I tried to get a lot more than we got."

Bob was in an expansive mood. There has been a lot of good news lately for the Heath Company and Zenith Data Systems. For example, Ellerton mentioned the contract whereby Heath/Zenith is selling 6,000 Z100s to the Air Force, Navy and Marine Corps over the next three years. (For further information, see *Sextant* #8, Winter 1984, or issue 77 of *Buss: The Independent Newsletter of Heath Co. Computers*.) He also mentioned that advertisements for Zenith computers have begun appearing on national TV. "The changes around the Heath Company, just the personal changes, since the announcement of the Air Force contract" and "the national Z100 commercials on TV ... have done a lot for the morale of the entire building. And believe me, the computer being built today is going to be a better computer for it. It won't stop."

Morale may have improved at Heath, but some things never change: the company is still holding firm to its policy of not discussing future products. In the question and answer session that followed his prepared remarks, Ellerton warned the CHUGCON audience that HUG is "walled off" from information about future products. "We've put ourselves in a unique position at HUG. We don't want to know, because *Buss* will tell us anyway."

As usual when Heath users gather, a HUGgie started things off by asking when we would see the long-awaited version 3.0 of HDOS. The gist of Ellerton's answer was "we're working on it." Then a few people asked questions about the mechanical details of the HUG discount program, and Ellerton assured them that Heath would implement the program in an equitable fashion.

The most interesting question of the evening came from Mike Cogswell, CHUG's Z100 hardware expert. He has a record of seeming well-informed about the

---

young men of the Milwaukee area, tapping illegally into computer systems.) How do we deal with young people who have the ability to learn quickly and yet do not have the supervision to give them proper direction?

Maybe there is a lesson here for us to learn, also.

Sure, these computer "tappers" are making their way into "no trespassing" areas. Yes, they are showing us that our present computer security systems are inadequate. However, a faulty security system is not the issue.

It obviously is not right to allow sabotage in order to find weak coding. This may show us that improvements, quick improvements, are needed to implement high level security systems. If young experimenters playing with remote computers are successful in tapping defense systems, then who knows who else may be accessing those systems? But then espionage is not the issue, either.

What is the issue?

We have a responsibility to do something before this type of action takes place. *We* must set up a line of communication with these young people.

How can we do that?

First, we need to be aware that this type of playing around is *not* validated. Second, we need to understand that the "kid next door" may be the *next* young person trying to access a computer system illegally.

With these two thoughts in mind, we must strive to point and *help* young people to stay within legal boundaries. We must *encourage* them to find areas of interest which produce constructive results, not *de*structive results. We must try to *involve* them in design projects, which result in building better computer systems. But most of all we must *direct* them by building in them an upright character.

Help, encouragement, involvement, and direction—we can only do these if we are communicating with them.

Having investigated both extremes: computer illiterates, on one hand, and those who know "too much" for what they are capable of controlling, on the other, we can see that there are people of all ages and of various computer expertise levels who need our help and direction.

When the *S.S. Titanic* struck an iceberg on her maiden voyage in 1912, James Kruch of Chicago was on board. He survived this harrowing experience. Then, when German submarines torpedoed the *S.S. Lusitania* in 1915, Kruch was again on board a doomed ship. For the second time he was rescued, surviving two great sea tragedies. Yet years later, when he was crossing a shallow stream, Kruch collapsed and drowned. This man, who had survived two ocean ship sinkings, lost his life in less than one foot of water.

Berkeley was an explorer, mountain climber, and big game hunter. He had shot lions and tigers and other dangerous animals, chalking up quite a record for himself. Yet he died, not by the claws of a charging lion or the cunning stealth of a man-eating tiger, but from the scratch of a barnyard cat! Blood poisoning set in and killed him.

Blondin made crowds gasp and applaud when he walked over Niagara Falls on a tightrope. He was perhaps the greatest tightrope walker who had ever lived. Few feats of daring had been seen to compare to it. He was hailed as the "Conqueror of Niagara." Yet Blondin's death was hastened by his tripping over a doormat outside his own house!

The point: "little things" can make a difference! Little things can have a big effect on those around us ... including anyone who uses computers.

If it is truly our desire to help others, we must place ourselves at their level of understanding. We must show them we can relate to their misunderstandings and misfortunes. We must show them that we care that they learn. Then we can begin to communicate with them.

My charge to you as a group and to you as individuals is to watch for, and to be open to, any computer user that you come in contact with, and guide them toward a better and clearer understanding of their computer and the responsibility that they should accept toward other computer users. That person could be a quiet visitor who drops in on a CHUG meeting—it could be the retired grandparent who lives just down the street—or it might be the neighbor kid who bugs you about playing games on your computer.

If *we* do not take the initiative to communicate with these people in understanding the "whole picture" and in understanding their responsibility to other computer users . . . *who will*???

inner workings of Heath and Zenith Data Systems.

"What about the Z3?" Cogswell asked. First, Ellerton pretended that he'd never heard of the Z3 ("who?"). Then he read a rather cryptic statement he'd prepared on the subject of future products: "the hardware gurus are guru-ing away."

What's all this about gurus and Z3s? Well. . . we've heard talk that "Z3" is the internal code name for Zenith's next computer, and there've been persistent rumors that Zenith is working on an IBM PC "clone."

Sending them away happy

The exhibitors donated about fifty door prizes, which were given away at the end of the dinner banquet. (A few were also given away at lunch.) Most of the prizes were worth between $25 and $100, and there were only 240 people in the banquet room, so a lot of people went away happy. The most valuable prize was a pair of 8" disk drives, donated by Floppy Disk Services. As chance would have it, the drives were won by Sam Yates, one of the primary organizers of CHUGCON 83. Judging by the hearty applause, most CHUG members thought this was an appropriate reward.

Organizing CHUGCON

After the conference was over, I talked to some of the organizers of CHUGCON. Mike Supley was the chairman of the CHUGCON steering committee, and is now the president of CHUG. I asked him if he had any advice for people who might be interested in organizing their own regional conferences.

First, Supley said, you should have a general theme for the conference. In the case of CHUGCON, the theme was educational and scientific interchange about Heath/Zenith computers. Second, you should start planning early. Third, you should establish a steering committee, and pick chairmen for various subcommittees.

To make everything work, you need to have "very regular" meetings of the steering committee. "We did a lot of telephone talking back and forth." But you don't want to get good communications at the price of individual initiative and delegation of authority. Supley thought that the chairman should not try to be "the person with all the ideas." Rather, the chairman should try to "coalesce the ideas" of the group.

"Draw from the strengths of the people," he said. "You should give the subcommittee chairmen the authority they need to execute their jobs."

I also talked to Sam Yates, the chairman of the vendor subcommittee. It's extremely important to persuade vendors to attend, Yates said, because the vendor area is responsible for drawing 80% of those who attend. The most difficult task is communicating to vendors that it will be worthwhile for them to attend. You need to be able to deal with vendors in their own terms, on a basis of mutual respect. "Relevant personal contact is your key." Yates ran up at least $100 in phone bills while organizing the vendor area for CHUGCON 83.

Was all the work worth it? Mike Supley noted that "over 600 people with technically oriented interests" came to see "state-of-the-art technology." Supley's theme for the conference was "educational advancement." By that criterion, CHUGCON 83 was undoubtedly a success.

Yates, like everyone else I talked to, agreed that the

conference was a success, and enjoyed his part in it. He admitted, however, "I don't think I'll be vendor chairman next year. I'd kind of like to enjoy the fruits of my labors."

Will there be other regional conferences?

CHUG has demonstrated that it's possible for a local users' group to put together a good regional users' conference. Now it's up to other HUGs to follow CHUG's lead. Perhaps, by the time you read this, someone will have organized a West Coast HUGCON, or a Florida HUGCON.

Stay tuned to this station for news of CHUGCON 84.

Additional Information

Capital Heath Users' Group
P.O. Box 2653
Fairfax, VA 22031

Heath Users' Group
Hilltop Road
St. Joseph, MI 49085
616/982-3463

Participating Vendors

ACCESS
P.O. Box 1053
Norristown, PA 19404
215/272-4807

ACT Marketing
104 Lakefront Drive
Hunt Valley, MD 21031
301/628-0260

Apogee Software
P.O. Box 15124
Savannah, GA 31416
912/925-3765

Cherry Engineering
P.O. Box 216
Laurel, MD 20707
301/498-4236 (8-10 p.m.)

Cleveland Codonics, Inc.
P.O. Box 45259
Cleveland, OH 44145
216/243-1198

CompuTech
P.O. Box 2027
Poquoson, VA 23662
804/868-8055

Evryware
P.O. Box 5204
Olympia, WA 98503
206/459-2395

Extended Technology Systems
1121 Briarwood
Bensalem, PA 19020
215/752-4604 (evenings)

Floppy Disk Services, Inc.
741 Alexander Road
Princeton, NJ 08540
800/223-0306

GROFFics
Route 4 Box 515
Chapel Hill, NC 27514
919/942-0650

H & H Computer Enterprises
P.O. Drawer H
Blacksburg, VA 24060
703/552-0599

H/C Forum
35 West Wainwright Drive
Poquoson, VA 23662

Heathkit Electronic Center
6201 Richmond Highway
Alexandria, VA 22303
703/765-5515

Hoyle & Hoyle Software
716 South Elam Avenue
Greensboro, NC 27403
919/378-1050

Husker Systems
of Nebraska, Inc.
4517 North 61st Street
Omaha, NE 68104
402/451-4988

Interface Technology
P.O. Box 745
College Park, MD 20740

Linton Printers
4600 Waverly Avenue
Garret Park, MD 20896

Micromation
9104 Red Branch Road
Columbia, MD 21045
301/730-1237

New Generation Systems
2063 Fort Lee Street
Herndon, VA 22071
703/476-9143

New Orleans General
Data Services
7230 Chadbourne Drive
New Orleans, LA 70126
504/241-9495

Newline Software
P.O. Box 402
Littleton, MA 01460
617/486-8535 (evenings)

Northwest Digital Systems
P.O. Box 15288
Seattle, WA 98115
206/362-6937

One Way Micro Systems
P.O. Box 2546
Reston, VA 22090
703/860-3307

Powerline Systems
P.O. Box 97
Lincroft, NJ 07738
201/747-2063

Robins, Inc.
8304 Hilltop Road
Fairfax, VA 22031

Schumacher Associates, Inc.
12619 Valleywood Drive
Woodbridge, VA 22192
703/494-7885

Sextant Publishing Company
716 E Street, S.E.
Washington, DC 20003
202/544-0900

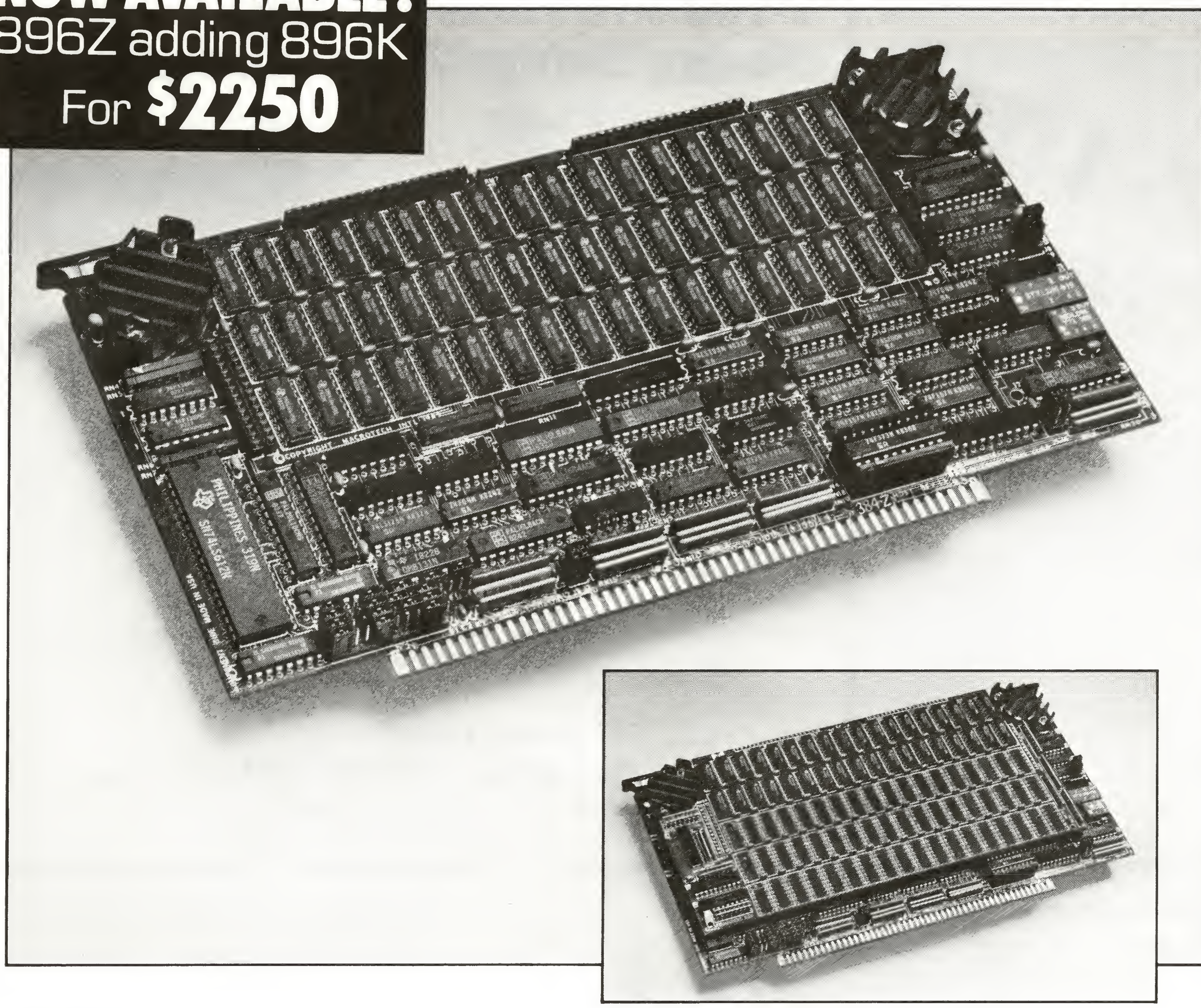
T & E associates, Inc.
P.O. Box 362
Millersville, MD 21108
301/987-4748

Technical Micro Systems, Inc.
P.O. Box 7227, Dept. H
Ann Arbor, MI 48107
313/994-0784

Terminals Unlimited
350 South Washington Street
Falls Church, VA 22046
703/237-8666

Trionyx Electronics, Inc.
P.O. Box 5131
Santa Ana, CA 92704
714/830-2092

Zeducomp
P.O. Box 68
Stirling, NJ 07980
201/755-2262



Why Won't All That IBM-PC Software Run on My Z100?

The operating system ought to be the bridge between the application program and the computer. Too many programmers use the PC's ROM as a shortcut.

Graham Wideman

People deciding which computer to buy will probably weigh heavily the amount of software available to run on each machine. Those considering the H/Z100 may well be encouraged by the fact that it appears to have many points of compatibility with the IBM Personal Computer. Both the Zenith Disk Operating System (Z–DOS) and the PC Disk Operating System (PC–DOS) are variants of the Microsoft Disk Operating System (MS–DOS). The standard presumption most of us make is that programs written for a given operating system should run on any computer that runs the operating system. From this, many will conclude that the huge PC–DOS software base may be tapped by the H/Z100.

From a hardware perspective, moreover, the H/Z100 appears to have been designed with compatibility considerations in mind. It has not one but two central processor unit (CPU) chips. Its 8-bit Intel 8085 processor allows it to run software written for the CP/M operating system. Its other processor is the 8088, the same one as the PC's.

Users who already own an H/Z100 may also find compatibility an important subject. Some have attempted to run some PC software. More often than not, however, they have encountered the infamous WILD INTERRUPT message, and failure of the program.

The lure of that PC–DOS software pool is not going to go away. Nor, I am afraid, are at least some of the obstacles that stand between the software and the average H/Z100 user. Some of the problems, however, can be overcome. When they are, a number of popular PC programs become usable on the H/Z100.

In this article, I'd like to examine the reasons behind the problems. In doing so, I'll describe a software package that we (Wideman Computer Consulting) have produced to emulate some of the functions of the PC. Called IBEm, it combats several of the PC-compatibility problems. Other problems remain. The problem of compatibility, however, has always faced computer users. It's not insurmountable. I hope this article encourages some of you to work on it.

Similarities and differences

There are two main areas of compatibility between the PC and the H/Z100. First, as noted above, the disk operating system is the same. Moreover, the disk formats are the same; so each machine can read the other's disks. And thereby, programs and data can be readily transferred from one machine to the other.

> *A large proportion of "runs under PC–DOS" software actually circumvents the operating system.*

This *should* tell all. To the computer purist, the function of an operating system is to ensure that applications programs have a standard interface to the particular computer on which they are running. Thus, we would reason that any differences between the PC and H/Z100 hardware are taken care of by MS–DOS.

The application-program interface provided by MS–DOS on each machine is indeed the same. However, the catch is that a large proportion of "runs under PC–DOS" software actually circumvents the operating system and accesses features of the PC by other routes, routes which are unique to the PC. (Different routes are used to accomplish the same goals on the H/Z100.)

So our next step is to examine the PC's "other routes."

How a PC programmer programs

To aid in this discussion, Figure 1 provides a picture of the conceptual layout of software and hardware in the PC. The arrows show the "proper" ways in which the various hardware features of the PC may be accessed—in other words, through PC–DOS. In order for an applications program to access the keyboard, it calls the operating system; the operating system finds out which key was struck and passes it back to the applications program.

For those readers interested in the actual MS–DOS "calling mechanism," it is implemented via "software interrupts" of the PC's 8088 central processor unit. Such an interrupt is an assembly-language instruction, INT xx.

A software interrupt is simply a glorified subroutine call, with one important difference: the address of the subroutine is specified in a four-byte vector in a fixed location in lower memory. This location is determined by the number xx of the interrupt (zero to 255).

As an example, most MS–DOS functions are accessed via INT 21H (hexadecimal), whose vector is at 84H. (For CP/M users, this is analogous to your CALL 5.) This procedure is used so that the applications program doesn't need to know where a routine is in memory, just which one it wants. MS–DOS fills in the actual addresses in the interrupt's vectors.

As I mentioned, PC–DOS isn't the only way to access PC hardware. But why go any other way? The answer to this key question is simple. In a number of cases, PC–DOS actually stifles much of the capability of the PC's hardware. A prime example is the lack of flexible video output available through PC–DOS. It is possible to send ordinary characters to PC–DOS and have them print on the screen. However, there is no way to perform often-needed terminal functions, such as moving the cursor, deleting to the end of the line, and so on. That means that a word processor can't do fancy video maneuvers through PC–DOS.

(I might note in passing that the H/Z100 *is* able to perform such func-

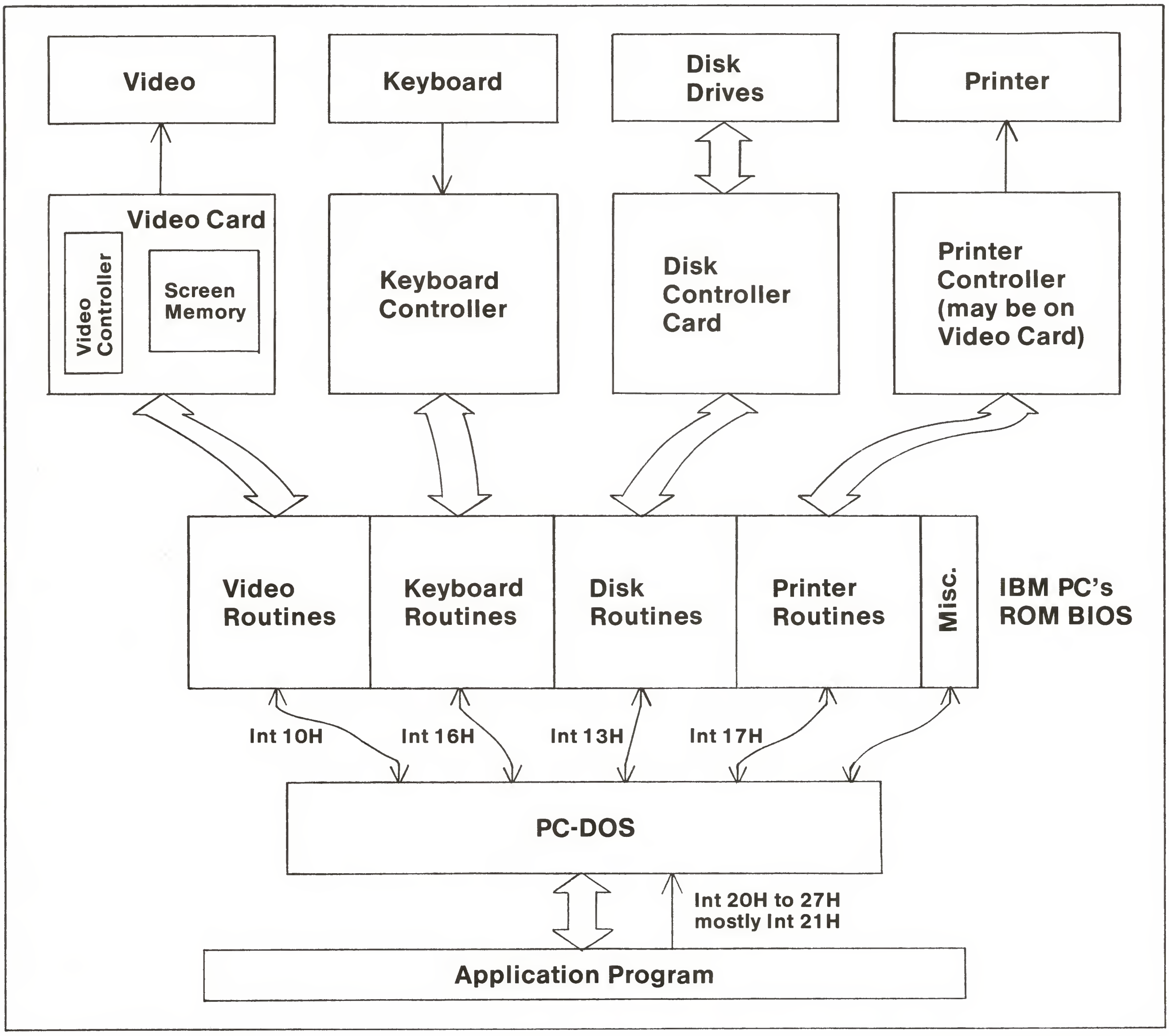


Figure 1: How an application program interacts with the IBM PC. What is *not* shown here is that many programs bypass PC-DOS and use the ROM directly; and other programs may write to and read from the video memory directly.

tions through Z–DOS, because the Z100's screen-handling software is set up to emulate the older H/Z19 terminal. These functions are performed by using ESCape sequences.)

Fortunately (or unfortunately, from an H/Z100 user's point-of-view), there are other easy ways to access the PC's hardware. These are mostly through the PC's read only memory (ROM) and the ROM's extensive basic input/output system (BIOS)—"basic" meaning rudimentary.

The PC's ROM "operating system"

The PC has a large ROM. It's large enough so that it even includes BASIC. And it contains extensive routines for accessing and controlling almost every feature of PC hardware; including disks, video screen, keyboard, printer, and cassettes.

In fact, I suspect that it was the decision that the PC be able to work with cassette which obliged IBM to put such an extensive BIOS into a ROM in the first place. (The usual procedure is to have much less in ROM, since changing the chip is cumbersome and expensive if revisions need to be made later.) The routines the PC has in ROM would normally be loaded from disk. Such a procedure would be impossibly slow if cassettes were the system's mass-storage device.

The PC's ROM routines are also accessed via software interrupts—interrupts having numbers which do not conflict with PC–DOS's, of course. In fact, PC–DOS uses many of these ROM routines itself. This situation is also shown in Figure 1.

So, as I mentioned, PC–DOS doesn't provide PC programmers with everything they need. Therefore, in order to write a sophisticated-looking package which can take full advantage of the PC's capabilities, many PC programmers have been obliged to use the ROM-BIOS routines directly.

And every time this happens in a PC program you are running on your H/Z100, you get a WILD INTERRUPT message.

So now we need a more detailed discussion of exactly which routines are probably causing that message.

Popular "PC misbehavior"

What follows, then, is a list of the PC-features which we found are often accessed through the ROM. It also gives details showing how attractive the ROM routines are relative to using PC–DOS—which should, in turn, indicate how frequently these are used by programmers. (I will also include comments on the H/Z100 emulation of each of these functions provided by our package IBEm, version 1.2.)

Video: This is interrupt 10H. The PC's ROM provides extensive routines for handling text on the screen, and for handling graphics. As noted, PC–DOS provides only simple, teletypewriter-

Circle #137 on Reader Service Card

Circle #219 on Reader Service Card

Circle #145 on Reader Service Card

like output to the screen. (So there is no direct cursor positioning, nor many other functions needed by word-processors.) Consequently, PC–DOS is almost never used in this regard by software attempting any sophistication.

The ROM provides fairly complete text handling, including positioning on the screen's X and Y axes and window scrolling. Routines are included for both of PC's video boards (monochrome and color-graphics boards) and several different display formats.

However, the ROM's graphics routines are not, in our opinion, complete enough to be useful by themselves. So our IBEm package instead provides routines to emulate operation with the PC's 80 x 25 monochrome text-only board.

We decided not to support the ROM's graphics and color, since the PC's screen format is inherently different from the H/Z100's, and the graphics routines are almost always circumvented by programmers. Not supporting color is not often a problem, because most software will adapt to run with either of the PC's video cards, and IBEm tells the software that a monochrome card is installed.

Although IBEm will not permit the running of programs which perform pixel-graphics functions, version 1.3 of IBEm *does* provide the PC's graphics *characters*. These prove to be quite useful in cases where software menus use a picture, such as an arrow character, to illustrate the keys.

They should be asking, "How compatible is it?" rather than just, "Is it compatible?"

I must note here that the data to be displayed on the PC's screen gets stored in memory within the PC's address space. Therefore, the PC programmer has the alternative of accessing the PC's video cards directly. This is almost universally done for pixel graphics. It is also done sometimes for text, when the programmer needs extra speed.

It is important to realize that this direct access to the display memory cannot be emulated in software on the H/Z100. So neither IBEm nor any other emulator package can cope with such programs. More on this subject later.

Another consideration which is sometimes inconvenient is the speed of programs operating under IBEm. While a particular PC–DOS program may operate, its video operations will always be slower than on a PC.

The reason for this is that in certain respects the PC's text handling is optimized in terms of how much time is consumed in putting a character on the screen. As a result, programmers have sometimes taken liberties with this speed: they have crammed more screen manipulation into their PC software than they would normally attempt on other machines, particularly terminal-based ones. When IBEm intercepts these commands, it then has to perform them on the somewhat less-optimized-for-text H/Z100.

Keyboard: Using the PC ROM's Interrupt 16H, the PC programmer can get full information on all the PC's keys, including the function and cursor keys, and the complete extra "case." (The PC has uppercase, lowercase, control characters, and an "alternate" case.) Although all the keys are reported via PC–DOS, the special keys (anything non-ASCII) are passed back in a rather inconvenient form (a "0" followed by the key code). Programmers have tended to prefer the one-call-gets-it-all routine found in ROM. (We implemented the ROM's INT 16H in IBEm so that we could emulate all of these keys on the H/Z100 keyboard.)

Printer: Although it is quite possible to access the printer through PC–DOS, some programs use INT 17H of the ROM for this function, notably at least some versions of MicroPro's WordStar. (This function is implemented in IBEm.)

Equipment report: Interrupt 11H on the PC is a function which returns information as to the amount of memory on the main board, number of disk drives, printer, video card, etc. (This is implemented in IBEm.)

Memory check: Interrupt 12H reports back the amount of memory available. (Implemented by IBEm.)

Disk access: Surely the main function of a disk operating system is to control the disks? So no one is going to circumvent PC–DOS on this matter, right?

Well, unfortunately, they do. Given fairly simple commands, the ROMs pro-

Some words on IBEm

As I have noted in this article, IBEm provides an emulation of many PC features on the H/Z100. Notable programs which this allows running are PC versions of dBASE II (version 2.3, which we use for our invoicing), WordStar (3.24), Perfect Writer (and Perfect Speller, version 1.0), and many less well known programs.

On the other hand, there are many programs which still will not run. The PC version of Lotus 1–2–3 uses direct screen access, while PC-VisiCalc has a simply unbelievable copy-protection scheme. (It actually deactivates any debugger which might be used to trace the cause of the problem!)

Unfortunately, we cannot even come up with a list of what runs or doesn't run, because programs change from version to version. This is exemplified by WordStar's version 3.3—in which the programmers have gone to direct access of the video board, no longer using the ROM routines. In any case, most of the big-name packages will become available for Z–DOS. Users will be trying to use a package like IBEm with the less well known ones (and with those expensive ones "borrowed" from friends with PCs).

We feel that although IBEm is not a panacea, it takes the H/Z100 a long way (as far as software can) in giving PC–DOS software a chance to run. If you are considering purchasing a copy, we suggest you try it first, if possible, at your local Heathkit Electronic Center.

The IBEm package includes manual, documented source code, and related utilities. Updates may be obtained by returning the original disk with a self-addressed, stamped return envelope. Heathkit Electronic Centers have been provided with evaluation copies; and some of the stores have it available for purchase.

vides rather powerful control over the disks, including reading from or writing to specific sectors. This allows programmers to copy-protect their software; here's one way they might do so.

You can write a small program which can be run just like any other, and show up in the directory with the name of your "real" program. This, however, will only be a "loader" program. You will put the real program directly onto the disk without including any directory entry for it. The loader program will know where to look for it. The loader will bring in the valuable real program from tracks near the center of the diskette, using the ROM's direct track-sector read functions.

So your valuable program is on the disk in an area which the operating system regards as empty. Then you record some bad sectors on a track just outside your valuable program.

What does this all do? Try to copy it. The file-by-file copy program (COPY) will copy only programs which are in the directory. The track-by-track copier (COPYDISK) will get stuck at the bad track. Valuable program protected! Now, this is only a rather simple-minded scheme, and not terribly difficult to circumvent with ingenuity, but it is often used. Unfortunately, this scheme precludes use of the software on the H/Z100 since the PC ROM's disk routines are not available.

For this reason, IBEm implements the most-needed track-sector read and write routines. These routines are also used by software which wants to use the disk without the DOS directory, or needs to go directly to specific locations for other reasons. As one result, we were most delighted to be able to use Peter Norton's PC disk utilities on the H/Z100. (These utilities allow viewing and patching disk areas, and recovering erased or damaged files. But see more on this below.)

It should be noted, however, that IBEm's disk routines cannot cope with non-standard sector sizes and numbering. This is because it is necessary for IBEm to work through Z–DOS's disk functions, in order that Z–DOS shall always know what is going on. Consequently, more sophisticated copy-protection schemes preclude use of the software on the H/Z100.

BASIC and miscellaneous routines

IBEm does not deal with compatibility between the PC's BASIC and the H/Z100's Z–BASIC. BASIC programs initially appear not to be compatible when transferred from the PC to the H/Z100 and vice versa. However, this is just because the two BASICs "tokenize" (convert keywords to single-byte codes) differently. This can be circumvented by SAVEing the program on the source computer with the ASCII option set. (See the BASIC manual on this.) This saves the program as text, which the other BASIC will swallow properly. There may still be problems if a program uses commands not available (or having different syntax) on the other BASIC. Some graphics commands will fall into this category.

There are two ROM routines we have not implemented on IBEm. These are the timer routine and the RS232 serial-port handler. The first we omitted because we haven't seen any software which uses it. (This is probably because PC–DOS has its own time call.) The RS232 function definitely *is* used, typically by "Terminal" packages. We may implement it in the future; but we have not yet turned up a package for test purposes which uses that function and does *not* do direct video. Suggestions from readers who come across such a terminal package will be welcomed. (When used under IBEm, it would currently get an "Unimplemented Interrupt 14H" message.)

How compatible can the H/Z100 be?

A package like IBEm allows the use of many more PC–DOS packages than the bare H/Z100 can run. However, there still remain serious obstacles to complete no-worry PC compatibility.

The major obstacle is graphics, and direct access to the PC's video memory for either graphics or text. In our opinion, there is only one real solution to this problem; and that is actually to make an S–100 board which, to software, looks like one of the PC's video boards. (It turns out that this would not conflict with the H/Z100's memory or port usage.)

If that were done, then it would be necessary to write routines (analogous to IBEm) which operate that screen memory board *instead* of the H/Z100's screen. This is not trivial, since a PC–DOS program might put things on the screen alternately through PC–DOS, the ROM, and by direct access. So both the ROM interrupts and Z–DOS would have to be implemented for this new video board.

We have performed some experimentation in this area. Our method was to place an ordinary S–100 memory board at the address of the PC video card. Then we wrote a routine that transferred the entire screenful of characters to the H/Z100's video memory. The routine does this each time the applications program receives a key from the console. (This strategy is used because programs get keys after they have finished doing things on the screen.)

Using this technique, we were able to run as an example the Norton Disk Utilities previously mentioned. However, this is *only* an experimental solution, since it works only if direct access is the only method the applications program uses to write on the screen. Additionally, copying the screen for every keystroke makes the keyboard response impractically slow. That is something you can live with in Norton's utilities, since you are usually making single-key selections from menus. It would be useless for word processors. (These experimental routines are provided with the IBEm package.)

Our next step is to investigate the practicality and usefulness of such an

Circle #116 on Reader Service Card

Circle #151 on Reader Service Card

Circle #145 on Reader Service Card

S–100 PC-video board. It seems to be a fairly straightforward task to interface a PC video card to the S–100 bus. This will be our strategy for performing tests. (Readers interested in this progress should contact us, since this article was written a couple of months before you read it.)

There will always be some packages which use other, more obscure features of the PC directly, and hence will not run on the Z100. However, I believe we will see a tendency toward sticking to PC–DOS or the ROM routines if programmers want their software to run on machines that are almost PC-compatible. And that is a family of growing size.

Who's to blame?

When discussing the matter of PC-compatibility, it is not very long before accusations start flying as to who advertised what degree of compatibility, and why it isn't there. The implication exists that the H/Z100 is compatible with the PC. It is also clear that most buyers are ignorant of the fact that compatibility comes in degrees. They should be asking, "*How* compatible is it?" rather than just, "Is it compatible?"

We are very sympathetic with Heath/Zenith users because the equipment comes so close. And in fact, the H/Z100 is a better machine (in our opinion). Unfortunately, many users don't realize the widespread practice of circumventing MS–DOS. This is all the more sad because it is actually the "runs-under-PC–DOS" packages which have contributed to customers' misunderstandings. These packages more accurately should state something like "runs under PC–DOS and requires the PC's ROM." The fact that such a program can actually be seen in the directory on an H/Z100, and then crashes when run, makes customers angry with Heath/Zenith, not with the software writers.

Whether or not one is understanding about the incompatibilities, however, there is still a benefit to be had from owning an MS–DOS machine. And that is that there are a lot of programmers writing for PC–DOS. As the PC-"compatible" market grows, these packages will become available for the other machines. The same could not be said for a machine based, say, on a Z8000 processor.

Other "compatibles"

This brings us to the topic of other PC-"compatible" machines. If you are considering the purchase of a PC-compatible, you should carefully check the compatibility claim. Based on the discussion here, you now know that, to be PC-compatible, the machine must provide PC-like ROM calls. (Although they need not actually be in ROM, nor at the same location in memory.) It must also provide video memory which looks just like the PC's.

You will find this combination elusive, and even the information is difficult to elicit, with misinformation abounding. For example, a reviewer for *Byte* magazine, looking at a recently released PC-compatible portable, stated that he thought it would run more software if a PC video card were stuck in one of its PC-compatible expansion slots. Oh yes? And where are the software interrupts to operate that board? And how do you switch between that machine's own video and the PC-card?

As you can see, the Elusive-Compatibility Problem is not just Heath/Zenith's. And irrespective of it, the H/Z100 remains a machine with many praiseworthy features.

Ordering Information

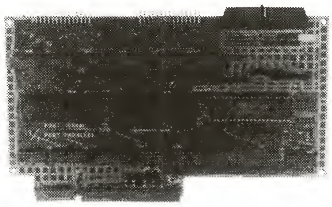
IBEm, $79.95 plus $4 shipping and handling to U.S. and Canada, $12 elsewhere; California residents please add 6% sales tax.
Wideman Computer Consulting
1320 Pepper Villa Drive
El Cajon, CA 92021

The Norton Disk Utilities
Peter Norton
2210 Wilshire Boulevard
Santa Monica, CA 90403

H/Z89 PERIPHERALS from SECURED COMPUTER SYSTEMS

PORT SERIAL
2/3 CARD I/O
PORT PARALLEL

"... not your typical vanilla-flavored serial and parallel interface ..."



Features:

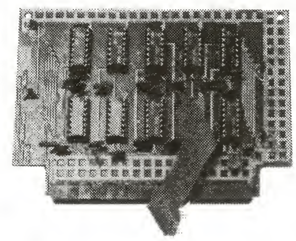
Chip independent design • Reduces computer data buss loading from 3 to 1 • Choice of Centronics or Epson parallel drivers for HDOS or CP/M • Complete documentation and installation instruction

- 2 Serial Ports
- Supports: Ring Input, External Clock, Auto Dialer
- 3 Port Parallel with 2 Level Interrupt Control
- Fully compatible with all models of H/Z 88, 89, 90 using CP/M or HDOS.
- Fully tested, 90 day warranty, two serial cables and a parallel cable (internal to computer) and software driver.

PRICE $199.00
Shipping & Handling $10.00

16K RAM EXPANSION CARD

Expands your H/Z89 RAM Memory capacity to a **FULL 64K!**



Fully compatible with:
H/Z 89 • H/Z 88
Magnolia Microsystems
CP/M and disk drive I/O interface cards

NOW INCLUDING SUPPORT MOUNTING BRACKET

Featuring:
Complete installation instruction • 90 day Warranty
Field reliability record now entering its 21st month
Now Only $65.00

HDOS Is a registered trademark of Heath Company
CP/M is a registered trademark of Digital Research

Shipping & Handling $5.00

PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
MAIL ORDER: 12011 ACLARE, CERRITOS, CA 90701 (213) 924-6741
TECHNICAL INFO/HELP:
8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930
Terms and specifications subject to change without notice.

Secured Computer Systems

ZENITH data systems SERVICE CENTER

Circle #157 on Reader Service Card

H/Z-100 COLOR GRAPHICS SOFTWARE

Want to explore **Z-BASIC** color graphics the easy way? Try these packages from **MICROSERVICES.** All you need is a color configured computer with 128K memory, a color monitor, and **Z-BASIC.**

ZANIMATE will help you rapidly draw and paint each frame in an animation sequence. Select frame sizes, vary location and time between each frame. Draw the background too, with the graphics editor **$64.95**

ZPALETTE, the original H/Z-100 palette program, contains a graphics editor for point-plotting images which can be painted in 92 hues! File your images for access by other **Z-BASIC** programs, and transfer imagery from one file to another.
NEW PRICE **$59.75**

ZPATTERN helps you determine the quality of your color monitor and if it needs maintenance.
NEW PRICE **$24.95**

Include 3% handling/postage ($2.50 minimum) — California add 6½% tax. Send check or money order to:

MICROSERVICES
P.O. BOX 7093
MENLO PARK, CA 94026
PHONE: (415) 851-3414

Circle #209 on Reader Service Card

ARE YOU PREPARED FOR TAX TIME?

TAXTYME will assist you in preparing and calculating your 1983 Federal Income Tax forms. **TAXTYME** simplifies filling out the forms by leading you through a series of formatted questionnaires. **TAXTYME** includes formatted questionnaires for IRS Forms 1040, Schedules A, B, and C. Schedule C is processed separately so that several Schedule C business activities can be calculated and combined on Form 1040. Information from Schedules D, E, F, and the many numbered Forms are calculated separately and entered on the questionnaires. Forms G, W, and 2210 are calculated automatically when required. **TAXTYME** makes decisions on Income Averaging, Marital Deduction, and penalty Form 2210 if necessary. **TAXTYME** then processes the information you enter, performs the necessary calculations, and produces a computer printout for direct transfer onto IRS forms. Requires a printer.

ONLY $59.95!

CP/M version for H89, H8, requires CP/M and MBASIC
ZDOS version for Z-100, requires ZDOS and ZBASIC

Newline Software

P.O. Box 402, Littleton, MA 01460 (617) 486-8535

ORDER NOW . . . SHIPMENTS BEGIN JANUARY 15, 1984

NAME ____________
STREET ____________
CITY ____________
STATE ____________ ZIP ____________

CHECK ONE
☐ H89, H8, CP/M
☐ Z-100, ZDOS

Send me ______ TAXTYME program(s) at $59.95 each.
Check one: ☐ payment enclosed ☐ send COD (add $3.00)
Send order to:
NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460
Foreign orders: add $3.00 Airmail, $10.00 for non-U.S. checks

CP/M is a trademark of Digital Research, Inc.
MBASIC is a trademark of Microsoft, Inc.
ZDOS, Z-100 are trademarks of Zenith Data Systems, Inc.

Circle #145 on Reader Service Card

Circle #110 on Reader Service Card

Circle #116 on Reader Service Card

MMM	EOX	ALS	AMR	GAO	PEP	KO	ARC	TX
5¼	10s35½	19¼	3s3½	18	4s35	3s49¼	49½	38

Circle #140 on Reader Service Card

Screen Dump!

Don't let your computer be overwhelmed by an avalanche of characters when the screen is transmitted.

Siebert Ickler

The terminal screen of your computer often contains a lot of useful information. At least I pretend mine does. There are many times when it would be extremely helpful to have that screen image saved. It helps in debugging programs to have a picture of what actually happened when the program went awry. The exact error message is often necessary in communicating with vendors. It saves time and prevents errors in sending patch listings to *Buss* or to your friends. Often, it would be handy to save the output of the operating-system utilities. Most of them only list their information on the console, which is the terminal screen on both the H/Z89 computer and the H8 with H/Z19 terminal.

Also, the H/Z89 and H/Z19 each have a set of 33 graphics characters which can be used to create pictures and charts on the screen. But the operating systems and utilities supplied by Heath make no use of them and provide no simple access to them for printing on paper or saving on disk. The only suggestion in the Heathkit manual is to write a BASIC program to PRINT on screen the appropriate control characters required to make a picture such as the American flag. This method is unsatisfactory for children and other impatient artists.

The terminal, however, comes with all the facilities required to move the cursor around the screen and to allow you to create a picture or graph. If you press the OFF LINE key, you can use the chart in the manual as a guide to the appropriate escape sequences and graphics characters and create a pretty picture or impressive chart on the screen. But wouldn't it be nice to be able to save that image and be able to display it later? If you could save the images you had created, they could be used later, as menus for programs or as part of a game.

This article will tell you how to capture the contents of the screen of your H/Z89 or H19. It will present working programs for capturing the screen image, saving it in memory, printing it, and saving it on disk. This leaves you with a skeleton around which you can build your own program—such as a one-screen picture editor or a dump-to-printer exit for the operating system.

However, this is all done in assembly language—which makes the program more difficult to write. You may wish to refer to the examples in the Heath and operating system manuals for help; and there are many useful articles in previous issues of *Sextant*. (A brief introduction to assembly language appeared in issue #8, Winter 1984, "Getting Into Z-DOS Assembly Language—Quietly," by D. C. Shoemaker. There is also "Disk Programming *Without* HDOS" by Richard E. Smith in issue #1, Spring 1982. And you may want to take a look at "An Introduction to CP/M for Assembly Language Programmers," by J. C. Hassall in *Sextant* #5, Spring 1983.)

To follow the text exactly, you must have the CP/M operating system, running on an H/Z89 or H8-H/Z19. You will have to be able to enter a program with an editor, assemble the result using the CP/M ASM utility, and load the result with CP/M's LOAD program. You will also need DDT, a standard CP/M utility program used to debug and patch programs. We will use it to examine memory. The CP/M utilities are all documented in the CP/M manuals provided by Heath when you purchase CP/M.

The program is fast enough to keep up with the terminal and it turned out not to be particularly difficult.

Hints on converting these programs for use under the Heath Disk Operating System (HDOS) are provided near the end of this article. But those of you running under HDOS will have to have a greater level of expertise than the user of CP/M. Under HDOS, you will not only have to know how to enter a program and assemble it, but also how to write a program to open and close a device driver or file and write to it. The HDOS information is contained in Heath's Software Reference manuals for HDOS and the HDOS System Programmer's Guide.

How did I find out?

Like a lot of you, I built my '89 from a kit. Sure, I had a lot of practical uses in mind, like word processing and writing software to sell. Perhaps I could even invent the next program which would sweep the field like the 'Calcs have. But I also wanted it just to have a computer. I work with large mainframe computers and the thought of being able to actually own a computer was irresistible. And, of course, I bought it to learn. And since I wanted to learn everything I could about it, I read the manual from cover to cover.

I don't know if this has happened to anyone else, but my kit came a unit at a time over several weeks. So I had plenty of time to read the manual. Because my disk drive came last, I played with the computer as best I could, which mostly meant learning about the terminal and its features. I managed to find a use for most of the escape sequences, but the transmit-page function (ESC #) escaped me. The little note that the computer needs a special routine to use this feature sounded like a challenge.

The transmit-page command causes the terminal portion of the H/Z89 to send the contents of its memory, which contains an image of the screen, to the computer portion of the H/Z89. It sends an exact copy of the screen with the appropriate escape sequences embedded to enter and exit reverse video and enter and exit graphics as required by the screen contents. It transmits from the upper left corner of the screen, sending all 80 characters, including blanks, on each of the 24 lines. It sends only one carriage-return character—at the end of the transmission—and beeps once at the end. So it sends a mimimum of 1,921 characters.

This seemed like a nice feature to use. As I said, I often would like to save the contents of the screen (or print them). Moreover, I could envision a simple full-screen editor which used all the features and keys of the H/Z89.

So I set out to transmit a page to a program. First, I tried it under HDOS, in Benton Harbor BASIC. I printed the sequence (ESC #) and tried to read the characters. I quickly concluded that it was designed to sound the speaker. What I was hearing, of course, were the warning beeps from the operating system trying to tell me that its console input buffer was full.

This is because the console input to the operating system (under both HDOS and CP/M) is invoked by an interrupt generated by the console it-

self. So when the terminal portion of the H/Z89 starts sending, it pretty much takes over the whole machine. It interrupts the computer's console input routine to send its next character to the buffer almost as soon as it finishes transmitting the last one. This does not leave time enough for a BASIC program to read the characters from the buffer.

Normally, BASIC wouldn't even try to read the characters at all until it got a complete line delimited by a carriage return. I had, however, thought of that and had instructed HDOS to send the characters immediately by POKEing the HDOS console mode byte (location 8406) to indicate character mode (value 129). This location and its values are described in the HDOS System Programmer's Guide.

And I used Benton Harbor BASIC's character input function (CIN) to try to get them as fast as they were sent. It wasn't fast enough, however. And since the buffer typically holds about 100 characters, and the terminal sends at least 1,921, the buffer is quickly filled. Then you get to listen to 1,800 or so warning beeps in a row. Since they come so close together, it sounds like one long tone.

Of course, I hadn't really expected this to work because of the note in the manual about needing a "special routine." I suspected this meant that I would have to use assembly language.

The first assembly-language routine I tried was to change the console-interrupt location to jump to my program, instead of to the operating system's interrupt-handling routine. The console-interrupt location is called UIVEC in the monitor listing provided by Heath with the H/Z89 computer. The monitor listing is in a section of its own and has instructions for usage. I have found it quite handy for learning about the H/Z89 and would like to insert a little plug here to thank Heath for including it and to recommend it to anyone who plans to really understand the H/Z89. There is a comparable listing for the H8.

By placing the location of my interrupt-handling routine in the UIVEC, my program should have received the characters (instead of the operating system receiving them). But I was unable to make this method work. Heath hadn't yet made the source code to HDOS available. And I didn't have the CP/M BIOS yet, so I had no interrupt-handler routine to study. Never having written one before, I must have done something wrong. My routines always failed to return control to HDOS.

In any event, I had already thought of an alternative method, which was to disable interrupts entirely and have my program read the console directly. I tried to do this in BASIC but again was unable to make it work. It probably failed because of BASIC's slowness. Or perhaps I POKEd the wrong locations because I hadn't correctly translated the split-octal notation used in the monitor to the decimal notation required by BASIC.

I was quick to go back to assembly language, anyway. I don't like PEEKs and POKEs, and assembly language was more fun to learn. And I finally succeeded! The monitor listing again proved particularly useful.

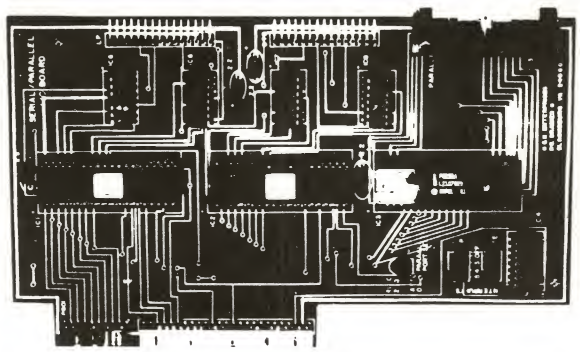
I patterned my console input/output (I/O) routines after the monitor routines called RCC and WCC (for read character from console and write character to console, respectively). The program is fast enough to keep up with the terminal and it turned out not to be particularly difficult. I have since included this code in programs written both for HDOS and CP/M to dump the screen. And I have used it as a one-screen picture editor. Since this method worked, I quit trying to find out why my interrupt handler didn't.

Sending the screen to memory

To understand the program better, look at Listing 1. This program, called FS, and all the programs in this article are for CP/M in 8080 assembler language. If you assemble this program, load it, and then run it, you will find that it appears to be quite useless. The screen will flicker briefly and then the bell will ring and you will return to CP/M.

However, if you examine memory, you will find the contents of the screen,

THE ORIGINAL SERIAL/PARALLEL INTERFACE BOARD



Replace the H88-3 to upgrade your system or use the KC1222 in place of it. Providing the two standard serial ports **PLUS** three parallel ports on one card, you have the best of both worlds. Use the parallel ports for printer, joysticks, instrumentation/control...KC1222 has all components including cables and hardware. KP1222 has the same items less the serial chips and cables.

PRINTER DRIVERS: The C.Itoh, Epson, Okidatas, Gemini and others are supported under HDOS and Heath's CP/M.

KC1222 Complete Kit	$128.00	PC1222 Bareboard	48.50
KP1222 Parallel Only Kit	97.00	Printer Drivers HDOS or Heath's CP/M	20.00

Printer Packages: We support the following printers as packages. We check out and set up the printer for your system. All you need do is plug it in and start using it! No more having to fuss with switch settings, making cables or any other aggravation. Included is HDOS or CP/M software, cable and checked out printer. Prowriter packages include 3K buffer upgrade.

Dot Matrix			
Prowriter Package	$440	Prowriter II Package	$720
Gemini 10 Package	390	Gemini 15 Package	585
Daisy Wheel			
Silver Reed EXP 500P Package	550	Silver Reed EXP 500P Package	795
C.Itoh A10-20 Package	595		
TTX-1014 Package	550		
C.Itoh F10 (55) Package	1525		

We also carry printer spoolers; contact us for particulars.

ZDS HARDWARE - PRINTER PACKAGES - SOFTWARE

MATHGEN: The keyboard math interpreter brings mathematics to life! Function roots, max and min, evaluate function integrals and first derivatives, hyperbolics, exponential forms, logs, trig functions, factorials, the list goes on! Define the functions from a file, batch processing style, or from the keyboard. Plot to the screen or your printer. CP/M or HDOS for only $40.00

DATAPLOT: Using the dot addressable graphics of your Prowriter, plot any definable math function in single or double density. Select axis size and scale, plotting quadrant, etc. Generate your data by MATHGEN, MBASIC or any other language. CP/M or HDOS for $25.00

PRN: A printer utility for the Prowriter, Epsons, and the H14. Print headers, footers, pause at page end, print multiple copies, left margin, printer mode and a host of other functions. CP/M or HDOS for $25.00

JS: Use your parallel ports for REAL TIME joysticks! Use up to four at a time with your favorite MBASIC or machine language keyboard game for HDOS. Program the response speed of the sticks, key sequence for different games, etc. Uses Atari joysticks. Only $25.00

dBASE II	$480	Friday	$245
Perfect Writer	280	Image	265
Perfect Clac	180	Lotus 1-2-3	400
Cache/Q	195	SuperSort	165
MITE	125	Benchmark V3.0	400
Jupiter (Power Line Systems)	145	Z-100 Screen Dump	25

This is but a partial list of the software we carry. If you need an item not listed here, call us - we either may already have it or can get it for you.

We fully support all of our independent-vendor products. If you have a question, we will **never** refer you to someone else.

As authorized ZDS dealers, we support the Heath/Zenith community with competitive prices and post sales support. Contact us for price and delivery on the Z system and accessories of your dreams.

VA residents add 4% tax. All of our products are shipped postpaid in the continental U.S. Otherwise, please add 5% for shipping. Foreign orders in U.S. dollars or drawn on a U.S. bank. Also includes customs house and address.

Group purchase discounts are available - contact us for details.

H & H Computer Enterprises, Inc.
P.O. Drawer H Blacksburg, VA 24060
(703) 552-0599

Circle #123 on Reader Service Card

MAIL MASTER®

— SPEEDS YOUR MAIL!

designed for use on ZENITH/HEATH Computers with at least 48K RAM Memory, using the CP/M Operating System*, one or more 5-¼" disk drives and printer. It will run on the Z100/H100 System with CP/M*85 and on the Z89/90 and H89/90 with CP/M*80. This package has four different programs (and easy-to-use booklet), as follows:

CREATE/COM — The first part of this program allows the user to enter the following data into memory: Special ID, Code Number, Names, Addresses, Telephone Numbers, etc. Program is displayed on the screen and is self-explanatory; when terminated, it provides a mailing list stored on the disk.

SORT/COM — The second part of this program sorts the newly-created mailing list in three ways:

[A] Alphabetic order, for a quick search of names.
[B] Zip code, for bulk mailing at low postal rates.
[C] ID code, for sorting special categories.

PRINT/COM — The third part of this program sends instructions to a printer for printing selected files of labels, with or without telephone number or special ID code, ganged-up or printed one label on each page.

EDIT/COM — The fourth part of this program permits search of specific records, correction and deletion.

Many similar programs are sold at a price three to four times higher and do not include half of the functions of ours. Our special price is now $59.95, with shipping. NY State residents add sales tax. Allow two weeks for delivery.

This package available in English (order S/N 083-E55-XXX); French (S/N 083-F55-XXX); and it will soon be available in Spanish (S/N 083-S55-XXX).

AT MSTC WE WORK FOR YOUR FUTURE AND OURS

THIS PROGRAM IS: **Interactive - Menu Driven - Powerful — and Uses Full Zenith/Heath Capabilities (i.e. 25th Line)**

M S T C
MICROCOMPUTERS SOFTWARE TECH CONNECTION
75-23 Main Street PO Box 111 Flushing, NY 11367
* Reg TM Digital Rsch Corp **(212) 263-9100, ext. 111**

Circle #159 on Reader Service Card

FREE SOFTWARE

PUBLIC DOMAIN DIRECTORY

NOW AVAILABLE ON HEATH CP/M 5¼ FORMAT DISKS.

CONTAINS MORE THAN 3,000 ENTRIES.

SUPPLIERS' NAMES AND ADDRESSES INCLUDED.

ASTRONOMY, AVIATION, BUSINESS, EDUCATION, ENGINEERING, GAMES, GRAPHICS, HAM RADIO, MUSIC, PROGRAMMING, TEXT EDITING, VOICE SYNTHESIS, UTILITIES AND MUCH MORE.

SHIPPED POSTPAID. MONEY BACK GUARANTEE.
SORRY, NO CREDIT CARD OR PHONE ORDERS. ALL 48 TPI:

☐ ON 3 HARD SECTORED SD $17.00 ☐ ON 2 SOFT SECTORED DD $12.00 ☐ ON 1 DOUBLE SIDED DD $7.00

HEAD
WARE

FREE BONUS UTILITY PROGRAM INCLUDED

2865 AKRON STREET
EAST POINT, GA. 30344

Name

Address

City, State Zip

Circle #192 on Reader Service Card

```
0100                    ORG     100H
00E9 =          CSTAT   EQU     0E9H    ;CONSOLE INTERRUPT ENABLE
                                        ;REGISTER PORT
00ED =          CTHRE   EQU     0EDH    ;CONSOLE LINE STATUS REGISTER
                                        ;PORT
00E8 =          CONS    EQU     0E8H    ;CONSOLE DATA PORT
001B =          ESCAPE  EQU     27      ;ESC CHAR
0100 =          START:  EQU     $
0100 DBE9               IN      CSTAT
0102 323E01             STA     INT     ;SAVE CURRENT STATUS
0105 AF                 XRA     A       ;ZERO A
0106 D3E9               OUT     CSTAT   ;SHUT OFF INTERRUPTS TO CP/M
0108 213F01             LXI     H,BUFFER
010B DBED       RL:     IN      CTHRE
010D E620               ANI     40Q     ;HOLDING REGISTER EMPTY?
010F CA0B01             JZ      RL      ;NOT READY YET
0112 3E1B               MVI     A,ESCAPE
0114 D3E8               OUT     CONS
0116 DBED       RL2:    IN      CTHRE
0118 E620               ANI     40Q
011A CA1601             JZ      RL2
011D 3E23               MVI     A,'#'
011F D3E8               OUT     CONS    ;TRANSMIT PAGE REQUEST IS
                                        ;ESC #
0121 DBED       INLP:   IN      CTHRE
0123 E601               ANI     1       ;CHAR READY TO SEND?
0125 CA2101             JZ      INLP    ;NOT YET
0128 DBE8               IN      CONS
012A E67F               ANI     177Q    ;STRIP PARITY? - COPIED FROM
                                        ;MONITOR
012C FE0D               CPI     15Q     ;CR SENT AT END OF SCREEN
                                        ;IMAGE
012E CA3601             JZ      ENDIN   ;IF CR GO TO END
0131 77                 MOV     M,A
0132 23                 INX     H
0133 C32101             JMP     INLP
0136 3A3E01     ENDIN:  LDA     INT
0139 D3E9               OUT     CSTAT   ;RESTORE CONSOLE INTERRUPTS
013B C30000             JMP     0       ;WARM BOOT TYPE EXIT
013E            INT     DS      1
013F            BUFFER  DS      5640
1747                    END     START
```

Listing 1: FS.ASM will send the contents of the screen to memory.

as it was just after you typed FS, in the area starting at location 013F (called BUFFER in the listing).

But first, let's discuss how it works.

Recall how transmit-page functions. When you send the transmit-page command (ESC#) to the terminal, it sends the image of the screen to the computer. This is the image as it appears, not as it was entered. It sends the image one character at a time starting at the upper left corner exactly as it appears on the screen—including blanks. There are no carriage returns or line feeds at the end of each line. Each line is exactly 80 displayable characters long and by counting the escape sequences (for graphics and reverse video) your program will be able to break it into lines, if desired.

One carriage return (octal code 15) is sent at the end of the entire screen. The terminal transmits the escape sequences to enter or exit reverse video or to enter or exit graphics mode as often as required to reconstruct the screen image.

In order to communicate to the console directly, we have to use the three ports defined at the top of the listing. These are: the interrupt enable register port; the line-status register port; and the console-data port, also called the transmitter holding register. (These are discussed in the INS8250 Asynchronous Communications Element description that was included with the H/Z89 documentation. In my documentation, it is an appendix to the H/Z89 operations manual.) Understanding how to use these registers and ports is where the monitor code comes in handy again. By examining the RCC and WCC routines in the monitor listing, we can see what to check for in these registers and how to go about it correctly.

Let's follow the program in Listing 1 in detail.

To follow the program, we will refer to the instructions by their starting location in memory. The locations are the four-character numbers in the first column of the listing. These are hexadecimal (H) numbers, to the base 16, so they run from zero to nine followed by A (for decimal 10) to F (for decimal 15). They indicate the memory location where the instruction or data will be stored.

Look first at the instruction starting at location 100 (labelled START:). The IN instruction there takes the value of the interrupt-enable register port and puts

it into the A register. The next instruction, STA, stores the A register into location 013E, labelled INT. Thus we have saved the current interrupt status before we alter it. (For more on interrupts, you might want to take a look at "Interrupt Programming Under HDOS," by Aaron Reynolds in *Sextant* #4, Winter 1983.)

The next instruction, XRA, zeroes the A register. We then send this to the interrupt-enable register port via the OUT instruction. This tells the INS8250 chip which controls the console port to disable interrupts. The instruction at location 108 points the HL register pair to the area of memory labelled BUFFER into which we will read the transmitted characters. The IN instruction at location 10B reads the value of the line-status register into the A register from the line-status register port. We want to examine bit 5 of this register to determine if the data register is empty so we can send our escape sequence to the console. We know which bit to check from the INS8250 documentation and the monitor listing. To do this, we use the instruction ANI (AND the byte immediately following the instruction with the contents of the A register). ANI 40Q (read: 40 octal) will set the zero status flag if bit 5 of the A register is zero. This is because the result of an AND instruction is a byte with ones in the bit locations where both operands have ones, and zeroes everywhere else. Since 40Q has a one only in bit position 5, we get a zero result unless the line-status register also has a one in bit position 5.

The JZ instruction at location 10F causes the program to loop until the Transmitter Holding Register is Empty (THRE, i.e., bit 5 is one). The instructions at locations 112 and 114 send the ESC to the console data port. We repeat this same type of loop in locations 116 through 11F to send the #.

Locations 121 through 133 are our loop to read the characters sent by the terminal. Locations 121 through 125 consist of looping until there is data ready in the transmitter holding register (data port). Bit 0 of the line status register is checked for a one to determine this (using the ANI 1 instruction). In location 128, we read the character into the A register. The instruction at location 12A is an ANI to zero the parity bit (bit 7). The instruction at location 12C checks to see if the character read was a carriage return (15 octal). If it is, all characters have been sent so there is a jump to the ending routine. If it is not, the program stores the character in the buffer (MOV M,A), adds one to the buffer pointer (INX H) and loops back to read another character (JMP INLP).

The ending routine (locations 136 through 13B) consists of restoring the interrupt-status register and returning to CP/M. We use JMP 0; this forces a

```
0100                    ORG     100H
00E9 =          CSTAT   EQU     0E9H     ;CONSOLE INTERRUPT ENABLE
                                         ;REGISTER PORT
00ED =          CTHRE   EQU     0EDH     ;CONSOLE LINE STATUS
                                         ;REGISTER PORT
00E8 =          CONS    EQU     0E8H     ;CONSOLE DATA PORT
0005 =          BDOS    EQU     5        ;JUMP TO CPM
001B =          ESCAPE  EQU     27       ;ESCAPE CHAR
0100 =          START:  EQU     $
0100 DBE9               IN      CSTAT
0102 327F01             STA     INT      ;SAVE CURRENT STATUS
0105 AF                 XRA     A        ;ZERO A
0106 D3E9               OUT     CSTAT    ;SHUT OFF INTERRUPTS TO CPM
0108 218101             LXI     H,BUFFER
010B DBED       RL:     IN      CTHRE
010D E620               ANI     40Q      ;HOLDING REGISTER EMPTY?
010F CA0B01             JZ      RL       ;NOT READY YET
0112 3E1B               MVI     A,ESCAPE
0114 D3E8               OUT     CONS
0116 DBED       RL2:    IN      CTHRE
0118 E620               ANI     40Q
011A CA1601             JZ      RL2
011D 3E23               MVI     A,'#'
011F D3E8               OUT     CONS     ;TRANSMIT PAGE REQUEST IS
                                         ;ESC #
0121 DBED       INLP:   IN      CTHRE
0123 E601               ANI     1        ;CHAR READY TO SEND?
0125 CA2101             JZ      INLP     ;NOT YET
0128 DBE8               IN      CONS
012A E67F               ANI     177Q     ;STRIP PARITY? - COPIED FROM
                                         ;MONITOR
012C 77                 MOV     M,A
012D 23                 INX     H
012E FE0D               CPI     15Q      ;CR SENT AT END OF SCREEN
                                         ;IMAGE
0130 C22101             JNZ     INLP     ;IF NOT CR GO BACK FOR MORE
0133 3A7F01             LDA     INT
0136 D3E9               OUT     CSTAT    ;RESTORE CONSOLE INTERRUPTS
0138 218101             LXI     H,BUFFER         ;GO TO START OF
                                                 ;BUFFER AND PRINT IT
013B 7E         PRLP:   MOV     A,M      ;GET THE CHAR
013C FE0D               CPI     15Q      ;CR ? MEANS END OF SCREEN
013E CA6301             JZ      ENDLP
0141 FE1B               CPI     ESCAPE
0143 3A8001             LDA     COUNT
0146 CA5301             JZ      ADDONE   ;ALLOW FOR ESCAPE SEQUENCE
                                         ;IN COUNTING
0149 3D                 DCR     A
014A 328001             STA     COUNT
014D CC6901             CZ      CRLF     ;OUTPUT CR AND LF IF ALREADY
                                         ;PRINTED 80 CHARS
0150 C35701             JMP     LPR      ;PRINT IT OUT
0153 3C         ADDONE: INR     A
0154 328001             STA     COUNT
0157 0E05       LPR:    MVI     C,5      ;BDOS CALL 5 - OUTPUT TO
                                         ;PRINTER DEVICE
0159 5E                 MOV     E,M      ;CHAR TO PRINT
015A E5                 PUSH    H
015B CD0500             CALL    BDOS     ;ASK CP/M TO PRINT THE
                                         ;CHARACTER
015E E1                 POP     H
015F 23                 INX     H
0160 C33B01             JMP     PRLP
0163 CD6901     ENDLP:  CALL    CRLF     ;HAPPY ENDING
0166 C30000             JMP     0        ;WARM BOOT TYPE EXIT
                ;
                ;               SUBROUTINE TO OUTPUT CR-LF TO PRINTER
                ;
0169 1E0D       CRLF:   MVI     E,15Q    ;CARRIAGE RETURN
016B 0E05               MVI     C,5      ;BDOS CALL 5
016D E5                 PUSH    H
016E CD0500             CALL    BDOS     ;ASK CP/M TO PRINT
0171 1E0A               MVI     E,12Q    ;LINE FEED
0173 0E05               MVI     C,5
0175 CD0500             CALL    BDOS
0178 E1                 POP     H
0179 3E50               MVI     A,80     ;RESTORE COUNT TO MAX
017B 328001             STA     COUNT
017E C9                 RET
                ;
017F            INT     DS      1
0180 50         COUNT   DB      80       ;# CHARACTERS IN A PRINT LINE
0181            BUFFER  DS      5640
1789                    END     START
```

Listing 2: FSPRINT will send the screen display to your printer.

CP/M warm boot but is the easiest way to return.

Testing FS

To verify the program's operation, we can test it as follows. First we assemble the program (using ASM) and then LOAD it as FS on the A: drive. DDT should also be on the A: drive. At the A> prompt, we type:

DDT FS.COM

This command loads DDT with FS under its control.

Next, DDT will give us a "–" prompt, at which we will type:

G100,13B

This runs FS, stopping at the JMP 0 instruction. At the "–" prompt, we type a third command:

D13F

to display the first 256 bytes in our buffer. Further D commands (D1F0, D2B0, D370, etc.) can be used to display the entire buffer. To exit DDT, type a CTRL-C.

Obviously, this is not particularly useful. This code must be made part of a program to make it more useful. Such a program might copy the screen to the printer, set a function key for screen dumping, copy the screen to disk, or be part of a screen editor or a subroutine for MBASIC or some other language to capture the current contents of the screen. We'll use it here to copy the screen to the printer and to disk. We'll need to add the code to take the screen image from the BUFFER memory area and send it to the printer or disk file.

This leaves you with a skeleton around which you can build your own program.

Sending the screen to the printer

Look at Listing 2, the program FSPRINT. The first part is the same as FS. We add the code from location 138 through 17E to print the contents of the BUFFER. The only change from the screen-reading logic of FS is to store the trailing carriage return so that we can use it as a marker in the print loop. The print loop basically takes each character to the printer from the buffer, adding a carriage return and line feed after each 80 characters.

Note the test for ESCAPE at location 141. This is done to make the count come out right if there were an enter or exit reverse video or graphics mode sequence. I send these to the printer in the example. Of course, this would create problems in actual use since the terminal and printer respond differently to given escape sequences. If you have an H/Z25 printer capable of handling the graphics, you should send the appropriate H/Z25 code instead of the one shown. If not, you should do something else when you see an ESCAPE—such as skip it (i.e., INX H twice and JMP PRLP) or refuse to continue (i.e., print a message and JMP ENDLP).

Try this program. As simple as it is you can use it as a screen copy program so long as you do not use reverse video or graphics. If you wish to capture the screen without the FSPRINT command being shown, and if you are sufficiently proficient with the escape sequences, you could use the OFF LINE key and the 25th line. That is, go off line and go to the 25th line; go on line and type FSPRINT; go off line and go to somewhere above the 24th line; and finally go back on line and type the return.

Sending the screen to disk

The program in Listing 3, FSDISK, copies the screen image to a disk file. The name of the disk file is typed on the command line just after the command FSDISK. Again (as in FSPRINT), the code from locations 100 to 12A is the same as the FS code to copy the screen to the buffer. Locations 138 through 19B are the code required to open the disk file in CP/M. The code in locations 1CA through 269 is that required to move the characters from the buffer to the disk buffer area maintained by CP/M.

In order to save disk space, the program does not move the trailing blanks at the end of each line to the disk buffer. This is done by looking ahead 80 displayable characters and then backing up to the first non-blank character. (See locations 22F through 247.) When looking ahead, the program maintains a flag indicating the status of reverse video (RVFLAG). This is because the trailing blanks are significant in reverse video and must be saved at the end of the line. That is the purpose of the code from locations 1ED through 214.

The subroutine at locations 26C through 29A does the actual move to the

Listing 3: FSDISK will save the screen display to disk.

```
0100                    ORG     100H
00E9 =          CSTAT   EQU     0E9H    ;CONSOLE INTERRUPT ENABLE
                                        ;REGISTER PORT
00ED =          CTHRE   EQU     0EDH    ;CONSOLE LINE STATUS REGISTER PORT
00E8 =          CONS    EQU     0E8H    ;CONSOLE DATA PORT
0005 =          BDOS    EQU     5       ;JUMP TO CP/M
001B =          ESCAPE  EQU     27      ;ESCAPE CHAR
005C =          FCB     EQU     5CH     ;DEFAULT FILE CONTROL BLOCK MADE BY
                                        ;CP/M FOR THE FILE ON THE COMMAND LINE
0080 =          DMA     EQU     80H     ;CP/M FILE BUFFER
0100 =          START:  EQU     $
0100 DBE9               IN      CSTAT
0102 32DA02             STA     INT     ;SAVE CURRENT STATUS
0105 AF                 XRA     A       ;ZERO A
0106 D3E9               OUT     CSTAT   ;SHUT OFF INTERRUPTS TO CP/M
0108 21E402             LXI     H,BUFFER
010B DBED       RL:     IN      CTHRE
010D E620               ANI     40Q     ;HOLDING REGISTER EMPTY?
010F CA0B01             JZ      RL      ;NOT READY YET
0112 3E1B               MVI     A,ESCAPE
0114 D3E8               OUT     CONS
0116 DBED       RL2:    IN      CTHRE
0118 E620               ANI     40Q
011A CA1601             JZ      RL2
011D 3E23               MVI     A,'#'
011F D3E8               OUT     CONS    ;TRANSMIT PAGE REQUEST IS ESC #
0121 DBED       INLP:   IN      CTHRE
0123 E601               ANI     1       ;CHAR READY TO SEND?
0125 CA2101             JZ      INLP    ;NOT YET
0128 DBE8               IN      CONS
012A E67F               ANI     177Q    ;STRIP PARITY? - COPIED
                                        ;FROM MONITOR
012C 77                 MOV     M,A
012D 23                 INX     H
012E FE0D               CPI     15Q     ;CR SENT AT END OF SCREEN
                                        ;IMAGE
0130 C22101             JNZ     INLP    ;IF NOT CR GO BACK FOR MORE
0133 3ADA02             LDA     INT
0136 D3E9               OUT     CSTAT   ;RESTORE CONSOLE INTERRUPTS
0138 3A5D00             LDA     FCB+1
013B FE20               CPI     ' '     ;CHECK FOR BLANK - NO FILE
                                        ;NAME TYPED
013D C27801             JNZ     CONTIN
0140 0E09               MVI     C,9     ;PRINT
0142 114B01             LXI     D,NOFILE
0145 CD0500             CALL    BDOS
0148 C30000             JMP     0       ;NO FILE NAME GIVEN SO QUIT
014B 4E4F204649NOFILE   DB      'NO FILE SPECIFIED ON COMMAND LINE -
                                QUITTING$'
0178 0E0F       CONTIN: MVI     C,15    ;OPEN
017A 115C00             LXI     D,FCB
017D CD0500             CALL    BDOS    ;TRY TO OPEN FILE AS IF
                                        ;EXISTS                →
```

How to Use: Zenith/Heath Computers, by Hal Glatzer

Introducing—the guide to your computer system! Written specifically for users, *Zenith/Heath Computers* is both an introduction to computing concepts for the novice and a reference tool for the expert. It also serves as a buying guide for new hardware and software. Clear presentations of technical information are supplemented with photographs and illustrations. This is not a complicated users manual!

Published in December 1982
S-A Design Books.

TABLE OF CONTENTS

THIS IS AN INDEPENDENT PUBLICATION PRODUCED BY S-A DESIGN BOOKS AND IS NOT IN ANY WAY ASSOCIATED WITH THE HEATH OR ZENITH DATA SYSTEMS CORPORATIONS.

Order your copy of *Zenith/Heath Computers* today for $19.95—including shipping and handling. If you're not completely satisfied with the book, just return it to us within 30 days for a full refund. Please indicate whether you would like us to ship the book by U.P.S. or Parcel Post, Special Handling. We can accept checks (in U.S. dollars, payable on a U.S. bank, please), money orders, Visa or MasterCard. To order, fill out the order card above and mail it with your payment of $19.95 for each copy to:

Sextant Publishing Company, Dept. S9, 716 E Street S.E., Washington DC 20003 • 202/544–0900

disk buffer (also called the DMA in this program), and calls CP/M to do the write to disk if the buffer is full.

The program determines that the buffer is full by looking at the last byte of the buffer pointer. (See the code at locations 274 through 276.) If this is zero, then the buffer is full because the buffer starts at location 80H and is 128 bytes long. So the next byte just after the buffer is location 100H (the start of the program). We could, of course, have counted the characters in the buffer until the count hit 128. But it is easier to take advantage of the location of the buffer in this case.

And finally, the code at locations 2BA through 2D7 is the program ending. It places a CTRL-Z (1A in hexadecimal) in the buffer to indicate end-of-file to CP/M (locations 2BA through 2B0) and then fills the rest of the buffer with zeroes (locations 2C6 through 2C9). The program then writes out the last disk buffer, closes the file, and returns to CP/M.

Some notes on HDOS

If you wish to use this technique under HDOS, you should find the modifications relatively simple. In FS, you should only have to change the ORG command to ORG USRFWA, and change the JMP 0 command to SCALL .EXIT.

In FSPRINT, you will have to replace the BDOS calls (in addition to making the FS changes). To do so, you will have to add an SCALL to open the printer device driver at the beginning and one to close it at the end. Instead of writing the print characters to the printer one at a time, you will have to format a 256-byte print buffer by moving characters from BUFFER to it (inserting line feeds every 80). This logic is similar to that used in the program FSDISK to move the characters to the DMA area. In HDOS, the end of each line would be delimited by a line feed only, instead of both the carriage return and line feed required by CP/M. (See the HDOS System Programmer's Guide for more details.)

Actually, some device drivers will handle the data one character at a time; so you might try setting BC to 1 and using the same logic as FSPRINT. That is, you would issue an SCALL to write to the printer for each character. If you don't care about speed, you could try putting a single character in front of 255 nulls (zeroes) and use the FSPRINT logic. All device drivers I am familiar with will merely ignore the nulls.

To convert FSDISK, you would again have to create your own 256-byte buffer and move the characters to it instead of to the DMA. You would have to use the appropriate SCALLS instead of BDOS calls to OPEN, WRITE, and CLOSE the disk file. And you would get the file name directly instead of using the FCB defined on 005C.

```
0180 3C                 INR    A        ;SO ERR 255 NO FILE BECOMES
                                        ;ZERO
0181 C2CA01             JNZ    CONT
0184 0E16               MVI    C,22     ;MAKE FILE
0186 115C00             LXI    D,FCB
0189 CD0500             CALL   BDOS     ;FILE DOES NOT EXIST SO
                                        ;CREATE IT
018C 3C                 INR    A        ;ERR 255 NO ROOM IN DIR
                                        ;BECOMES ZERO
018D C2CA01             JNZ    CONT
0190 119B01             LXI    D,NOSPACE
0193 0E09               MVI    C,9      ;PRINT
0195 CD0500             CALL   BDOS     ;PRINT ERR MSG
0198 C30000             JMP    0        ;GIVE UP
019B 4E4F20524FNOSPACE  DB   'NO ROOM ON DIRECTORY TO CREATE FILE -
                             QUITTING$'
01CA 21E402    CONT:    LXI    H,BUFFER
01CD 22DE02             SHLD   BPTRL    ;POINT NEXT LINE TO START OF
                                        ;BUFFER
01D0 218000             LXI    H,DMA
01D3 22E202             SHLD   DMAPTR   ;SET DMAPTR TO START OF
                                        ;BUFFER
01D6 2ADE02    NEXTLN:  LHLD   BPTRL
01D9 22E002             SHLD   BPTRM    ;SET NEXT CHAR TO MOVE TO
                                        ;START OF LINE
01DC 3E0D               MVI    A,15Q    ;CR?
01DE BE                 CMP    M
01DF CABA02             JZ     ENDLP    ;IF CR THEN WE ARE DONE -
                                        ;CLEANUP
01E2 3E50               MVI    A,80     ;# CHARS IN A LINE
01E4 32DC02             STA    COUNT
01E7 32DB02             STA    CHK
01EA 2ADE02    FNDLN:   LHLD   BPTRL
01ED 3E1B               MVI    A,ESCAPE
01EF BE                 CMP    M
01F0 C21702             JNZ    FNDLN2
01F3 3ADC02             LDA    COUNT
01F6 3C                 INR    A
01F7 3C                 INR    A
01F8 32DC02             STA    COUNT    ;ADD 2 TO COUNT TO ALLOW FOR
                                        ;ESC SEQ
01FB 23                 INX    H
01FC 3E70               MVI    A,'p'
01FE BE                 CMP    M
01FF C20A02             JNZ    FNDLN4
0202 3E01               MVI    A,1
0204 32DD02             STA    RVFLAG   ;ESC p found so set reverse
                                        ;video flag
0207 C32802             JMP    FNDLN3
020A 3E71      FNDLN4:  MVI    A,'q'
020C BE                 CMP    M
020D C22802             JNZ    FNDLN3
0210 AF                 XRA    A
0211 32DD02             STA    RVFLAG   ;ESC q found so reset reverse
                                        ;video flag
0214 C32802             JMP    FNDLN3
0217 23        FNDLN2:  INX    H
0218 22DE02             SHLD   BPTRL
021B 3ADB02             LDA    CHK
021E 3D                 DCR    A
021F 32DB02             STA    CHK
0222 CA2F02             JZ     BACKUP   ;AT END OF LINE BACKOFF FOR
                                        ;BLANKS
0225 C3EA01             JMP    FNDLN
0228 23        FNDLN3:  INX    H
0229 22DE02             SHLD   BPTRL
022C C3EA01             JMP    FNDLN
022F 3ADD02    BACKUP:  LDA    RVFLAG
0232 B7                 ORA    A        ;OR A WITH ITSELF JUST TO SET
                                        ;ZERO FLAG
0233 C24A02             JNZ    DISKIT   ;RV IN PROGRESS SO EVEN
                                        ;BLANKS COUNT
0236 2B                 DCX    H
0237 3E20               MVI    A,' '
0239 BE                 CMP    M
023A C24A02             JNZ    DISKIT   ;LAST NON-BLANK FOUND
023D 3ADC02             LDA    COUNT
0240 3D                 DCR    A
0241 32DC02             STA    COUNT
0244 CA5F02             JZ     CRLF     ;NO NON-BLANKS SO JUST PUT
                                        ;OUT CR LF
0247 C32F02             JMP    BACKUP
024A 2AE002    DISKIT:  LHLD   BPTRM
024D 7E                 MOV    A,M
024E 23                 INX    H
024F 22E002             SHLD   BPTRM
0252 CD6C02             CALL   DISKOUT
0255 3ADC02             LDA    COUNT
```

```
0258 3D                   DCR     A
0259 32DC02               STA     COUNT
025C C24A02               JNZ     DISKIT
025F 3E0D       CRLF:     MVI     A,15Q    ;CARRIAGE RETURN
0261 CD6C02               CALL    DISKOUT
0264 3E0A                 MVI     A,0AH    ;LINE FEED
0266 CD6C02               CALL    DISKOUT
0269 C3D601               JMP     NEXTLN
                ;
                ;         DISKOUT - SUBROUTINE MOVES A REG TO BUFFER
                ;                   &  WRITE TO DISK IF NECESSARY
                ;
026C 2AE202     DISKOUT LHLD      DMAPTR
026F 77                   MOV     M,A
0270 23                   INX     H
0271 22E202               SHLD    DMAPTR
0274 AF                   XRA     A
0275 BD                   CMP     L
0276 C0                   RNZ              ;NOT TO END OF BUFFER YET
0277 218000     TODISK    LXI     H,DMA
027A 22E202               SHLD    DMAPTR   ;RESET DMAPTR TO BEGINNING
027D 0E15                 MVI     C,21     ;WRITE SEQUENTIAL
027F 115C00               LXI     D,FCB
0282 CD0500               CALL    BDOS
0285 B7                   ORA     A        ;SET ZERO FLAG
0286 C8                   RZ               ;GOOD WRITE
0287 0E09                 MVI     C,9      ;PRINT
0289 119A02               LXI     D,NOROOM
028C CD0500               CALL    BDOS     ;WRITE ERROR MSG NO MORE
                                           ;ROOM ON DISK ETC
028F 0E10                 MVI     C,16     ;CLOSE
0291 115C00               LXI     D,FCB
0294 CD0500               CALL    BDOS     ;SAVE WHAT WE GOT
0297 C30000               JMP     0        ;QUIT
029A 4E4F204D4FNOROOM     DB      'NO MORE ROOM ON DISK - QUITTING$'
                ;
                ;         END OF DISKOUT SUB
                ;
02BA 2AE202     ENDLP:    LHLD    DMAPTR
02BD 3E1A                 MVI     A,1AH    ;CTL-Z MEANING END OF FILE
02BF 77                   MOV     M,A
02C0 23                   INX     H
02C1 AF                   XRA     A
02C2 BD                   CMP     L
02C3 CACC02               JZ      ENDFL
02C6 77         FILLZ:    MOV     M,A      ;LOOP TO FILL BUFFER WITH ZEROES
02C7 23                   INX     H
02C8 BD                   CMP     L
02C9 C2C602               JNZ     FILLZ
02CC CD7702     ENDFL     CALL    TODISK   ;WRITE OUT THE PADDED RECORD
02CF 0E10                 MVI     C,16     ;CLOSE
02D1 115C00               LXI     D,FCB
02D4 CD0500               CALL    BDOS     ;CLOSE FILE
02D7 C30000               JMP     0        ;HAPPY ENDING
                ;
02DA            INT       DS      1
02DB 50         CHK       DB      80       ;# CHARACTERS TO CHECK FOR A PRINT LINE
02DC 50         COUNT     DB      80       ;# CHARACTERS IN A PRINT LINE
02DD 00         RVFLAG    DB      0        ;1 IF RV ESC IN PROGRESS 0 OTHERWISE
02DE            BPTRL     DS      2        ;BUFFER NEXT LINE POINTER
02E0            BPTRM     DS      2        ;BUFFER NEXT CHAR TO MOVE/WRITE
02E2            DMAPTR    DS      2        ;NEXT LOC IN DMA FOR MOVE
02E4            BUFFER    DS      5640
18EC                      END     START
```

And finally . . .

This is the end of the promised programs. I did mention some other uses for the screen image in the beginning of the article. To make a print routine driven by a special function key, you would have to attach the program to the operating system. Then, whenever the special key you had selected was pressed, you would copy the screen as in the example programs. To make a simple one-screen editor, you would need code at the beginning of your program to echo what is typed on the keyboard to the screen, until the special key you had selected was pressed. You would then save the screen to disk or print it as in the example programs.

I have available for $7.95 an FS program for HDOS which will copy the screen to disk and/or printer. If you mention this article, I'll include the source code. It is in Z80 mnemonics for the Microsoft M80 assembler, unfortunately. So it is not exactly the same as in the listings here. Also, it cannot be assembled with ASM, which comes with HDOS.

This program has a couple of nice extra features for print control and storage format. I and my children have found it quite useful. My children use it as a general-purpose text editor. They can't type well enough to last for more than a screen, anyway. I have used it to save the screen when testing and to prepare patch listings. It can optionally store the disk file in a format compatible with the Ed-A-Sketch graphics editor from The Software Toolworks. And I have used it to modify the maze on their Munchkin game.

Ordering Information

HDOS screen-dump programs, $7.95
Available only in 5¼″ hard-sectored, single-density HDOS format.
Mention this article to request inclusion of source code in Z80 mnemonics for Microsoft M80 assembler.

Siebert Ickler
P.O. Box 1571
Corpus Christi, TX 78403

More on HDOS Stand-Alone

Squeeze more disk space from HDOS—automatically.

Charles E. Cohn

In *Sextant* #6, I described how users of the Heath Disk Operating System (HDOS) could get more usable disk space through stand-alone operation.

Stand-alone is an undocumented feature of HDOS. Briefly, it involves booting up with a standard disk containing all system files, and then switching (RESETting) to a working disk which only needs to have SYSCMD.SYS and PIP.ABS on it. This is particularly valuable for those with a single-drive system, but multi-drive users will also find it useful. (For more, see "Squeeze More Disk Space Out of HDOS," Summer 1983. By the way, this article was followed up by some excellent comments from Jack McKay of Washington, D.C., in the letters column of *Sextant* #7, Fall 1983.)

A drawback to stand-alone as I described it is that it is necessary—every time you boot up—to type in commands to load your device drivers, as well as the command SET HDOS STAND-ALONE. All that typing gets old after a while.

Fortunately, it is possible to eliminate this minor inconvenience by use of another undocumented feature of HDOS. Whenever HDOS is booted up, it tries to execute a program named PROLOGUE.SYS. If a program with that name is on the system disk, control will pass to it immediately. If it is not present, control will pass directly to the command processor.

Listing 1 will eliminate the typing. All you will have to do is to boot up with your system disk and put in the date. Within a few seconds, the computer will tell you to change to your working disk.

```
* Prologue to set up stand-alone operation
        XTEXT   HOSDEF
        XTEXT   HOSEQU
        ORG     USERFWA
START   EQU     *
*Load the line printer driver (omit if no line printer)
        LXI     H,LPNAME
        SCALL   .LOADD
        JC      ERROR
*Repeat the above for any other device drivers (except SY:)
*Load the overlays
        MVI     A,0                     Load overlay 0
        SCALL   .LOADO
        JC      ERROR
        MVI     A,1                     Load overlay 1
        SCALL   .LOADO
        JC      ERROR
*Clear channel -1, which the boot process leaves open
        MVI     A,-1
        SCALL   .CLEAR
        JC      ERROR
*Now change disks
        LXI     H,SYNAME
        SCALL   .RESET
        JC      ERROR
*Finally, pass control to SYSCMD.SYS on new disk
        MVI     A,0                     Normal exit
        SCALL   .EXIT
*Error handler
ERROR   MVI     H,7Q
        SCALL   .ERROR
        MVI     A,1                     Abort exit
        SCALL   .EXIT
LPNAME   DB      'LP:',0
SYNAME   DB      'SY0:',0
        END     START
```

Listing 1: Assembled, this program should be named PROLOGUE.SYS and placed on your boot disk. It will install HDOS STAND-ALONE automatically upon boot-up.

Using the program

To use the program, first assemble it, and name the resulting object-code program PROLOGUE.SYS. Then put it on your boot-up disk. You may have to customize the program for your system. If you don't have a printer, for instance, you should delete the coding that loads the printer driver. If, on the other hand, you have drivers for other peripherals (except SY:, which is always resident), put in analogous coding to load them.

(But what about those times when you don't want to go into stand-alone mode on boot-up? You may want to do something with the boot-up disk instead. You can manage that by using still another undocumented feature of HDOS. When the computer says ACTION <BOOT>, type I (for Ignore). HDOS will then not look for PROLOGUE.SYS, but will go directly to the command processor as soon as the boot is complete.)

One of the things you might want to do under stand-alone might be copying files with ONECOPY. It is helpful to be in stand-alone mode when using this utility. This allows you to return to normal operation when you are through instead of having to reboot. All you have to do is see that a disk containing SYSCMD.SYS is in place when you hit Control-D to get out of ONECOPY. If your source disk has nothing but data files on it, use the /MOU command to put in your boot-up disk before exiting ONECOPY.

If ONECOPY is on your boot-up disk, it is probably easier to let the prologue set up the stand-alone operation and then momentarily remove and re-insert the disk than to type in the stand-alone setup commands yourself.

Guide to Writing for *Sextant*

In which our inimitable technical editor offers some hints on style and content.

John Walker

Have you thought of writing about your computer? About what you do with it, and how?

Sitting there at home, you may figure that most people who write about computers are professionals: either computer professionals or professional writers. Well, while professionals do account for a lot of writing, a lot more is done by people who only consider themselves users or hobbyists. Their knowledge—and yours—is no less valuable because it focuses on one area. The problem you found challenging or fun may be tying someone else in knots. The program you've learned to use to its fullest may not have been reviewed yet.

Whether you're a hobbyist, a writer, a programmer, or a hardware specialist, you may very well be carrying around the idea for an article for *Sextant*. To get your knowledge to our readers, you mainly need: 1) a Heath/Zenith computer; and 2) a willingness to write about it.

***What is* Sextant?**

Sextant is the independent magazine for the entire Heath/Zenith computer community and for users of Heath/Zenith-compatible computers.

We're one of a growing number of magazines dedicated to covering a single brand of computer, its activities, and the applications of its equipment. We report on Heath/Zenith, we evaluate them, we show how their equipment and services can be put to work in consumer, recreational, educational, and business applications.

We also report on the entire industry that has grown up around the Heath/Zenith computers. This includes the producers of hardware add-ons. We cover the software, too. We cover all the products that are produced by independent suppliers to the Heath/Zenith community. Some of these companies have grown to the point where they're producing complete computers on their own. Since those computers are compatible with Heath/Zenith computers, we cover them, too. And we'll also include any new companies that produce similarly compatible equipment and computers.

So it's important to remember that we're an *independent* magazine. We're not employed by Heath/Zenith. We're paid by thousands of readers and dozens of advertisers. No one firm provides more than 2% of the gross income of Sextant Publishing Company. We can call the shots as we see them.

We're descended from *Buss: The Independent Newsletter of Heath Co. Computers*. Started in 1977, it's arguably the oldest independent system-specific newsletter in the microcomputer industry. *Buss* has become an indispensable support tool for users of Heathkit and Zenith computers. On the chance that you're new to this, by the way, they're the same computers: if you buy them assembled, they bear the Zenith Data Systems label. Put together from a kit, they'll have the Heath Company logo.

That Heathkit connection is important for *Sextant* and has been even more important for *Buss*: people who build their computers want to know a lot about them and their applications. Some *Buss* readers get involved with all the circuitry, details, and fine print of their equipment and software. Others want to keep up on the latest products available. For whatever reason, they tend to have a high degree of interest and involvement with their computers.

***Who reads* Sextant?**

With *Sextant*, we've moved into the field of those who are primarily concerned with *applications*. More and more, there are readers who know how to use their Heath/Zenith computers in a particular context and are seeking to expand their use and move into other areas. But they have more of an eye to getting a job done economically, rather than experimenting with their machines' potential just for the challenge of it or because they intend to be computer professionals.

This does not mean *Sextant* will ignore computer enthusiasts and professionals. Nor, for that matter, will *Sextant* duplicate or replace *Buss*. The major differences between them are those that are inherent to each medium. A magazine can cover issues at more length and in more depth than a newsletter can. A newsletter can give more quick announcements and act as a "party line" among readers. Depending on their level of knowledge and involvement, many readers will find it advisable to subscribe to both. In fact, the writers for *Sextant* are frequently readers of *Buss*. They write articles both for the new user and for others at their own level. They all need in-depth product reviews. They all want to know how their computers can be used more effectively. And they all like to know they have a source of detailed technical material that goes deeper than they need right now. *Sextant* has to cover the whole spectrum.

The substance of a *Sextant* article

What does* Sextant *want articles about?

Anything *specific* to Heath/Zenith equipment and users. Hardware, software, applications, your experiences with all the above, and all the possible interrelations. Practically anything is grist for the mill.

The key is that the article should show something users or prospective users can do with their equipment. That does not mean an exclusive concern with the latest software. But we do figure that our readers have a "show me" attitude. If it's an article on how to beef up your hardware, it should show who would benefit from the increased potential. If it's an article on some technically interesting feature, it could show why it's good to be aware of it, even if you have no need for it right now. People can grow into a need for certain knowledge. And part of the job of a magazine like *Sextant* is to keep them informed about what they'll run into down the road.

We're interested in general news and information articles, too. But they need a Heath/Zenith slant. If it's an article just on telecommunications, say, we probably can't use it. But if it's about someone's using Heath/Zenith equipment or related software to do the work, *we want to see it*.

Well, since you raised the point, what do you* not *want articles about?

Maybe we shouldn't have raised the point. If in doubt, *send it*.

But really, the main limitation concerns articles that do not refer to Heath/Zenith equipment, users, or applications. We figure our readers can get general information elsewhere. The general press covers general concerns well enough. But there's a lack of readily available information about Heath/Zenith in the general computer press. That's the gap we fill.

But, since we've opened that can of worms, and since writers do well to know of editors' prejudices, we *have* found that certain articles seem *harder to write* than others.

1. "My first six months with my computer" (or any other product). Here, the

problem is that it's too easy to get snowed under by your own experience. It becomes difficult to universalize it to be of use to someone else who's just starting out. At the other extreme, it becomes too easy just to give general information that could be gained from any user or sales brochure.

2. "Putting in the 235A/4 framis in place of the Z440 widget." You might want to write such an article because you had to learn the little monster pin by pin and want to spare others the grief. But how many others will want to pick up the 235A/4 at their local World War I surplus store? It must have some relevance to a fair number of readers. Remember, though, that lots of times, your wacko idea *may* have wide relevance, so see *Style*, below, to avoid some of the pitfalls of over-specialization.

3. "Here's a real good word processing program" or accounting program or what not, for you to type into your computer. The problem here is that the program you would most likely write yourself has probably attracted other people's attention, too. And been put up for sale. So readers come to your offering with a lot of heavy competition in mind. Too many of our readers are willing to pay for the convenience and reliability of purchasing a well-regarded program available on disk. Typing in a few hundred lines grows old, fast.

What sort of programs are you looking for?

Basically, games and utilities. They should be relatively brief and of general interest. We're looking for things that people do "to" their computers, rather than "with" them.

So we've had to turn away some very nice programs. This is because of our need to keep things specific to our readers. An energy-audit program, for instance, may be outstanding, but not everyone owns their home. We can presume, though, that everyone has a computer. So we stick to things that, theoretically, all our readers can use.

Do you only take short programs?

Yes and no.

If the main point of the article is just the program itself, then it should be kept short. In BASIC, 150 lines is a pretty good warning point that you're getting on the long side.

There's a lack of readily available information about Heath/Zenith in the general computer press. That's the gap we fill.

But we *will* take much longer programs.

With longer programs, though, the goal is not just to type the program into your computer to have a nice game or utility. With longer programs, the article has to be much more educational, so that readers learn about programming, or about some particular kind of software. The program is the example, the illustration, not the main attraction.

Submitting to *Sextant*

How developed should my article be before I submit it to* Sextant*?

That's up to you. We'll consider submissions at practically any level of development.

Query: just a letter asking us if we'd be interested in a given idea.

Abstract: a few paragraphs giving the basic points of the article.

Outline: the same information as an abstract, but it allows inclusion of smaller points without using up a lot of space. In general, we prefer "sentence-structure" outlines instead of "subject-heading" outlines. We need a sense of what you're going to say. So, "It's really hard" is better than "Relative Difficulty."

Rough: this could almost be a finished article, but you could leave out parts you're still working on, parts that need further research, or parts that you want our input on. Make sure to note such places clearly, though. Charts, illustrations, etc., could be rough sketches or photocopies or contact sheets.

Final: everything neat, tidy, and complete, including charts and illustrations in camera-ready form with captions.

Which is better?

That's for you to decide: which is better for *you*? If you have a number of ideas you're playing with and you're trying to decide, then a query, abstract, or outline would be best. If you really want to write about something in particular, and you're going to keep on trying until you get published, then a rough or final might be better.

The main difference for you will be in the timing: the closer it is to final form, then the quicker we can give you a firm answer; and, if we accept it, the sooner we can get it into print. But we hope to be around for a little while and we hope you will be, too. So putting an article off for an issue or two shouldn't be any crisis.

Of course, the earlier it gets printed, the earlier you get *paid*: how important that is, is also up to you. Starting out with a query may take a little time. But rewriting a rejected final also takes time. *Handle it in a way you find comfortable.*

In what form should it be?

Ideally, printed matter should be sent on plain white paper, 8½" by 11",

double spaced, with 1″ margins. Please don't send text, on *either* paper or disk, that's been right-justified, with extra spaces added to make all the lines the same length. If we accept the article, we'll only have to take them out. Nor should copy be all uppercase. It's hard to read and if accepted, we'd have to work on it to make it upper- and lowercase.

However, we're not dogmatic. As long as we can read it, we'll do so and at least give you some feedback on it. It's perfectly all right, for instance, to send in an article run out on fan-fold computer paper, without any form feeds at the folds.

Do I need to include a disk with my initial submission?

No. Generally, you need only send in the article on paper. The disk copy can be kept at home. It's only necessary to send it in when an article has been accepted.

The one exception to that rule is when you're submitting a short program where the program itself is the heart of the article. Then it has to be on disk. This is the only practical way we can test and evaluate the program. (Yes, we *do*.)

But you should have the disk copy in reserve. It makes copy editing infinitely easier than otherwise. And we use computer typesetting that reads the copy from disk.

What disk formats can you read? What word processor formats?

Practically any format that can be produced on a Heath/Zenith computer. If you have an H8 or H/Z89, we have a slight preference for HDOS over CP/M; on the H/Z100, a slight preference for Z-DOS over CP/M.

The only disk formats we have problems with are: 80 track/96 tracks per inch (under any operating system at all); and 5¼″ soft-sectored HDOS. Also, if you have an H11, that can be a problem. But *never* hold back from submitting an article because you don't have it on an "approved" disk format. We can usually find ways to get around limitations.

Similarly, we can handle practically anything your word processor can turn out—although we do have our preferences. If it's feasible, we prefer ASCII-only text files without any control characters. On screen, the file should look just like it does on paper. Some word processors (such as WordStar) set the high bit of the last character of every word, others put in control characters instead of carriage returns. We only have to take that stuff out so as not to confuse our typesetter.

As always, however, submit first and worry about technicalities later.

What about illustrations?

First, consider your article's potential for illustrations (even if you don't think it has any). For a product review, for instance, you might be able to get hold of a company's press photos. Even photos of disks and documentation can be useful.

If it's at all possible, photos, wiring diagrams, charts, etc., should be camera-ready. We have a limited arts-department capability, so it will help if you can produce "finished" graphics: india ink or other crisp black ink on white copy for drawings, etc.; large black and white glossies for photographs (8″ x 10″ glossies reduce well for printing purposes); or color slides. If in doubt, talk it over with us.

And if you use anyone else's material, *tell us.* Preferably, you should check with them, first. Most of the things you want to use (equipment pictures or schematics, say) are probably owned by folks who'd be glad to have them reproduced with credit. But copyrights are still copyrights and have to be checked long before deadline.

(Again, don't let that get in the way of submitting the article.)

Submitting an article to more than one publisher

***What if I'm submitting my article to someone else as well as* Sextant?**

Tell us. And tell them. You *must* let people know if an article is being looked at by another publisher. In publishing, it's a major no-no to publish something that's appearing someplace else *without both publishers' agreement.* What a writer usually sells is "first-time" publishing rights. And that "first-time" is what gives the work a lot of its value. Some publishers flatly refuse to republish something that has appeared elsewhere.

Sextant, however, has no hard and fast rules against considering an article submitted elsewhere. But joint publication and republication have to have the other guy's permission. Copyright problems aside, however, *Sextant* will only republish an article if there's little chance our readers saw the first printing. So choose where you want it printed most, and submit there first. Most publishers respond quickly enough that you don't need to sit around forever waiting for a yes or no. If you do sit around waiting, *check.* If needs be, drop a note to them withdrawing the submission.

What if you're the ones taking all the time?

Er... well, it does happen. If it seems as if we're taking a long time, feel free to drop a note or to call us on the phone. (Ask for John Walker, the technical editor. He's probably the culprit.)

Payment

How much do you pay?

We pay at or above the market rate for computer-oriented publications: from $50 to $100 per *Sextant* page of printed copy. Of course, that's a rough figure. It can be affected by a number of factors. You can't just pull out a ruler to check how much you were paid per page. Listings and tables get paid toward the low end. Actual text gets paid more. If your article took a lot more editorial work here at *Sextant*, that's going to lower the rate. Some writers will get paid more or less just depending on how great we think the subject, the writing, and the timeliness.

When will I get paid?

In the past (and currently), you would get paid *after* publication. Typically, you'd get two or three checks. The first would be the largest, going down from there. If you're a professional writer, you'll realize that's not typical in the publishing industry. We've done it that way for a couple of reasons.

First, it lets our editor sit down with an issue and go over all the articles at leisure. An article can be judged in the light of everything it contributes to that issue. Moreover, all the work is done: editing, corrections, production, graphics, last minute changes you noticed had to be made, and everything else that could happen.

Next, we have time to see what our readers think, through reader service cards, letters to the editor, and phone calls.

Of course, another reason for stretching out the payments has been good old *cash flow.* Frankly, we hope to be able to change things.

We expect to adopt a policy whereby you'll receive a small check as soon as the final version has been sent to the

typesetter; then the balance would follow upon publication.

Can I make arrangements for earlier payment?

Yes. If you want, we can settle on a price upon acceptance and pay immediately upon publication. In general, though, you'll get *less* than you would if you were willing to put up with our current, home-grown way of handling things. That's no guarantee, though. And we can understand that a professional writer, for instance, may prefer less money now to more later. We're assuming that most of our writers are writing as a sideline and the extra wait will be no real inconvenience. In any event, feel free to raise the question.

The goal, after all, is the same for all of us: the more you can offer us, the more we can offer our readers, and then the more we can offer you. It can be a very productive circle.

Style

What are your standards for grammar, punctuation, etc.?

Be careful. But don't get ulcers. Quite frankly, each magazine seems to accumulate its own little pet peeves. We have ours, too. But some of the industry's most cherished pets are little more than arbitrary dictates with no rhyme or reason to them. The rules for combining quotation marks with punctuation marks, for instance, were invented by typesetters on the basis of what they thought looked pretty. (Like I said, we all have our pet peeves.)

The chief punctuation rule is: does it aid in understanding the sentence? Write as you speak, more or less; so that commas indicate a pause; periods indicate moving to a new thought; paragraphs go to a new subject; and so on.

Don't let your sentences run on forever. The active voice is usually better than the passive. Sentences should be shorter than paragraphs—and paragraphs shorter than pages. A page of solid print is death.

Are there more important matters of style than just grammar, etc.?

You bet there are. Ideally, articles for *Sextant* should be written in such a way as to be interesting to the advanced reader, but still accessible to the beginner. And this affects the substance of articles. The four most important points concern remembering your audience: people with a show-me attitude who hope to put your ideas to work in their own environment.

1. Justify yourself.

Each article for *Sextant* should clearly indicate *why* a reader should bother to read the article. What good will something do? For whom?

2. Universalize your experience.

Try to figure out who can benefit from your article—and how. Don't think merely in the usual terms of business, consumer, hobbyist, etc. Think also in terms of people who can use a given capability. Some hobbyists are playing with communications networks similar to those used by giant corporations. Some consumers have bookkeeping needs more complicated than some small businesses. *Don't rule an audience out of consideration.*

3. Concretize your examples.

Don't be afraid to give concrete examples—particularly of the type that will be found in a variety of environments. "Human interest" and "human engineering" are often close cousins. The way it actually happened on your project may add a lot of life to a rule of practice you discovered or put to work.

4. Explain yourself—gracefully.

It may seem like a trivial rule, but it's an important one: always spell out on first writing. Whether it's "AT&T" or "RAM" or "AFRTS" or "NATO" or "ATC". The fact that everyone who reads *Billboard* will recognize the National Association of Theater Owners is irrelevant. Or that readers of the conservation press will recognize the Appalachian Trail Conference.

For more complex terms or for equipment, a similar attitude should prevail. You can't explain everything, but you can briefly indicate what you're talking about when you use a term. "The industry is agog over Heath's framis. It has established itself as the most advanced widget smasher around."

This explanatory material should appear in the *introduction* to the article, by

the way. Don't wait until after you've already raised an issue. More readers will be able to follow your article more easily with that material under their belt. And even if they decide the subject is too technical for them, they will still have learned something valuable.

If you're dealing with a lot of terms that the general *Sextant* reader might not be familiar with, you can always consider incorporating a glossary with the article, or a bit of background on the hardware. Material like this, set in its own box, not only explains things but it breaks up the page a bit.

How you apply these four rules is up to you. Sometimes, you need only flag a point at the beginning to show who should be interested in your new framis. Then you can go on and speak hardware for the rest of the article. Sometimes you will be talking almost exclusively in terms of applications, with the equipment taking a subordinate role. The main point is still to remember your audience: show them why your subject is of interest, let them get an idea of how they can use it in their world, and make it interesting and understandable to them in its own right, even if they're only observers.

If that's what I should do, what should I avoid?

One of the biggest things to avoid is assuming that the reader will have your level of knowledge. A game program is one case where you're likely to leave material out of the explanation. Don't—even though "everybody knows that." Since it may be played by someone who is not skilled at computers, it should be readable and playable by *them*.

Call in someone who knows little or nothing about computers and have them try out the game relying only on the instructions that show up on the screen. Then you'll see how much you've taken for granted. This is basically the process the Heath Company uses to test out its kit-building instructions. *It works.*

The experienced user will breeze over the "extra" information. But inexperienced users can become hooked on computers when they find they can use the little monsters all by themselves.

For tutorials, a similar rule applies. Try to get your article read by someone who is of the level of knowledge you're aiming it at.

Considerations like these, by the way, are why many writers bury a work for a while, whether overnight or for a month or so. You can come back to it with new eyes. "Coming back to it with new eyes" is a talent worth acquiring, by the way. If you dragoon your spouse or siblings as non-experts, after a while they become experts and useless to your writing efforts. (Competition, even.)

Philosophy

If you've read the style guides from other magazines, you may have noticed that we're a bit different. Editorially, this expresses itself in The One Absolute Rule: if in doubt, *submit*. You will not get a rejection because of grammatical boo-boos or something else that makes your writing not up to snuff. We figure it's our job to *edit*—that's what we're paid for.

As near as we can tell, it seems we're willing to spend a lot more time with an author and an article than some other magazines are. Some magazines are willing to print an article just as they get it. They leave it up to the readers to muddle their way through. Others expect perfection to be handed to them right out of your envelope.

Well, we figure we've found a good balance.

To a large extent, though, we don't have any choice. The community we represent and reach doesn't allow us to be either sloppy or smug. Most of you who write for *Sextant* are yourselves active users. You're probably not professional writers, but you can spot weaknesses in a technical article. That might be because you have a lot of knowledge and can see errors quickly. But it might also be because you're just starting out and can recognize when a writer has left you at sea. So as writers, you may need pointers because as readers you're demanding.

Getting writers and readers together—Heath/Zenith users communicating with other Heath/Zenith users—is *Sextant*'s job. With your help, we hope to do it better in the future.

Build a Sound/Clock Board for your '89

Now you can program music, sound effects, the time, and the date.

Ray Albrektson

My two children took one look at the latest "educational program" I had cooked up for them and sniffed with disdain. "It doesn't make any noises, Daddy!" I realized then that to compete with my son's Atari, my programs needed more than the standard "beep" provided by a stock H89. I would have to add (i.e., build) a sound board.

My first problem was to decide which sound chip to use. When I studied the data sheet on the General Instruments AY3-8910 Programmable Sound Generator, one of the most versatile such chips on the market, I was amazed. In one 40-pin package there are three oscillators (each digitally adjustable in frequency and volume), a noise generator (for explosion sounds), and an envelope generator that can modulate any of the other sounds with a variety of waveforms. Most intriguing of all, the AY3-8910 has two 8-bit ports that can be easily programmed for input/output (I/O) functions.

Since I was going to have to build a new board anyway, I also considered what else I might use it for. I recalled that I had read recently of a new clock-calendar chip from OKI Semiconductor, the MSM5832. This clever chip has an on-board oscillator and all the internal circuitry to provide data on time and date in a form easily adapted to acceptance by a microcomputer. A little bit of circuit juggling produced a clock/sound board design that interfaces both chips with the H89, and does it with only four additional chips and a few minor components.

The place you'd normally put such a board inside your computer is on one of the expansion connectors on the right side. But what if your right side is already full? With two disk controllers and a serial board, there is no room physically available for another accessory! Don't despair—with a little signal-stealing, you can mount the clock/sound board on the left side of the H89. However, build it and test it for the right side first, and when you know it is working, you can move it to the left side. (More on how to do this later.) The design discussed here should work for most users. You must, however, have available at least one of the three serial ports on your serial board.

P504		
Pin	1*	Ground
	2*	D0
	3*	D1
	4*	D2
	5*	D3
	6*	D4
	7*	D5
	8*	D6
	9*	D7
	10*	Ground
P510		
Pin	1*	+5 volts
	2*	Ground
	3*	A0
	4*	A1
	5*	A2
	6	Read
	7	Write
	8	Wait
	9*	I/O Serial 0 (active low)
	10	I/O Serial 1
	11	I/O Serial LP:
	12	I/O Cassette
	13*	2.048 MHz clock
	14	1.8432 MHz clock
	15*	Reset (active low)
	16	I/O 0 (bit D7 of port 362Q)
	17	I/O 1 (bit D8 of port 362Q)
	18	Interrupt level 3
	19	Interrupt level 2
	20	Interrupt level 1
	21	+12 volts
	22	−12 volts
	23	−5 volts
	24	+12 volts
	25	Ground

*Denotes pins used by clock/sound board

Table 1: Accessory bus signals for the H/Z89. The clock/sound board uses several of these.

Getting the big picture

The circuit I'm going to describe here is shown in Figure 1. The circuit has three major sections: address decoding (so the H89 can select the clock/sound board); sound generation and amplification; and the timekeeping section. Address decoding is taken care of by U1 and U2. The sounds are generated by the AY3-8910 chip (U3), and they are amplified by LM386 (U5). In addition, the AY3-8910 serves as a parallel interface to the MSM5832 clock/calendar chip. The timekeeping section includes: the clock/calendar chip (U4); capacitors C2 and C3; and crystal X1, which provides the time base. The hex inverter (U6) assists in reading the clock. The clock chip itself is powered by a 9-volt transistor-radio battery to provide uninterrupted timekeeping.

H89 I/O made simple

The H89's central-processor unit (CPU) board is arranged so that accessory boards (disk controller, serial interface board, etc.) plug vertically into the three accessory positions on the right side. The signals available from these plugs are shown in Table 1, and the sound/clock board uses several of them.

When the computer wants to make use of one of these boards, such as to communicate to a serial port on the serial board, it needs to select just one board in such a way that the desired board is activated and the others (such as the disk controller board) are unaffected.

The way that the computer selects each board is through I/O decoding inside the CPU board. Pins 9-12 on each of the CPU's right-side 25-pin accessory plugs provide decoded I/O strobes (a strobe is a single pulse) for use by accessory boards.

Pin 9 is designated I/O 0, and is low whenever the CPU does an IN or OUT operation at addresses 320Q through 327Q. ("Q" designates octal notation.) Note that pins 2, 4, and 5 on the accessory connector plugs carry address bits A0-A2. This means that each accessory board must have the smarts to figure out a specific I/O address from the addresses on the A0-A2 lines and the I/O strobes on pins 9-12 of the accessory bus.

For example, if an accessory board has eight gadgets on it, and these have I/O addresses 320Q-327Q, then to pick gadget 4, the CPU will pull pin 9 low to indicate that the address is in the range of 320Q-327Q while the gadget address (03Q) is indicated by the A0-A2 address lines. As a result, the accessory board would have to "decode" this information and know to activate gadget 4 when this combination was encountered.

For this project, I chose to use I/O 0 (pin 9), which provides for decoding addresses 320Q-327Q. I selected this address because this is the one available for the "middle" chip on the HA-88-3 serial interface board. If you use all

three serial ports of your serial board, you're out of luck. At least one needs to be vacant (with the 40-pin serial chip actually removed) for the clock/sound board to work.

How the decoding hardware works

The decoding is actually accomplished by U1, which is a 74LS138 "one-of-eight decoder/demultiplexer." This is an "active-low" device, which is great, since the H89 uses "active-low" strobes. The A0-A2 inputs on U1 are connected to the A0-A2 lines of the H89 address bus at pins 3-5 of the accessory plug. When an I/O address between 320Q and 327Q is selected, pin 9 goes low and enables U1. The address on A0-A3 then determines which of the eight outputs of U1 will be brought low. If the I/O address is 320Q, then pin 15 goes low; if 321Q, then pin 14 goes low, etc. This provides unique address decoding for up to eight devices. Since only three are used, the other five are available for other projects requiring I/O decoding. For instance, you could add another AY3-8910 chip for stereo sound effects!

Construction of the I/O decoder section

You have some choices in how you build this board. It is fairly easy to build it with wire-wrap materials. You can even make a perfectly nice wire-wrap board with a fine drill and a piece of phenolic or Formica. Personally, I figure it is much easier to use a printed-circuit board. Even for a one-shot operation, it may not be difficult to find a company that has the photographic equipment and sensitized board to produce a printed-circuit board from the camera-ready artwork in Figure 3. If that's impractical for you, you can order the printed-circuit board from me. (The rest of the parts can be purchased easily, either by mail order or at your local electronics store. See Table 3.)

It is usually a good idea to build and test new circuits in stages. First the I/O decoding will be built and tested, then the programmable sound generator section will be added and tested, and finally the clock/calendar section. Don't cut this process short! Over a hundred people have built this board, and not one of those who followed the step-by-step checkout method was unsuccessful.

First, check your tool inventory. Do you have a nice, light, and low-power soldering iron of the "Princess" type? Use fresh rosin-core solder, and remember that old solder does not flow as well as new (something about the rosin crystallizing). Have Solder Wick on hand to suck up surplus solder; and when it's all done, you will want some trichlorethane (flux remover) to remove the excess flux. At each checkout stage, you will need a volt-ohm-milliammeter (VOM). And buy, borrow, or beg a digital logic probe.

A word about static damage. The programmable sound generator (PSG for short) and clock/calendar chips are metal-oxide silicon (MOS) devices, and are susceptible to static damage. Keep them in their protective foam until just before installing them. When installing them, make sure that you are in a low-static environment, such as at a metal table on a wood or vinyl floor. (Not carpet!) Be sure you are not carrying a charge of static electricity before pick-

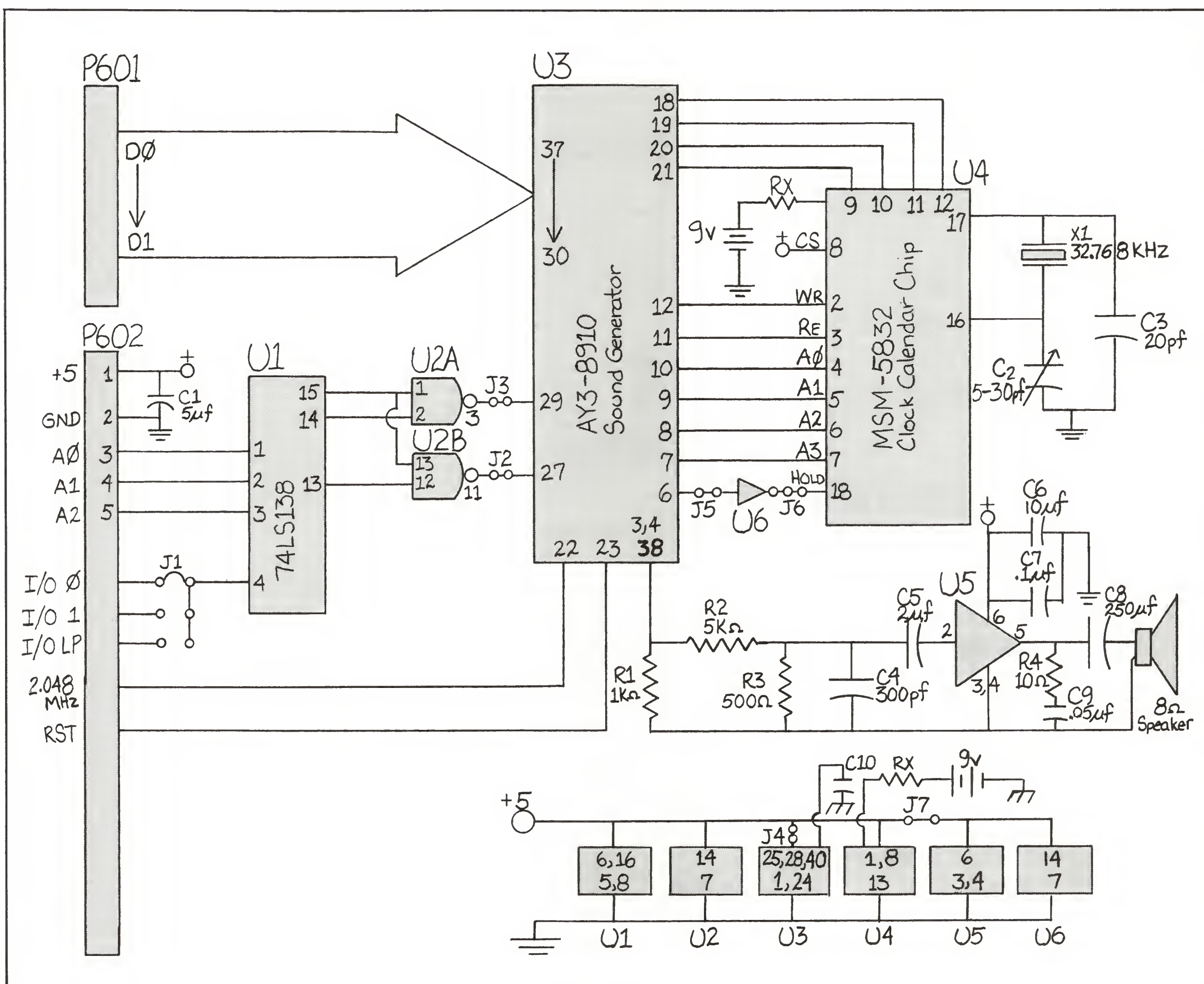


Figure 1: Circuit diagram of the clock/sound board. Address decoding is handled by U1 and U2. Sounds are generated by U3 and amplified by U5. The timekeeping section includes U4, U6, and crystal X1. The diagram at the bottom indicates the wiring of the six chips in relation to the +5 volt supply.

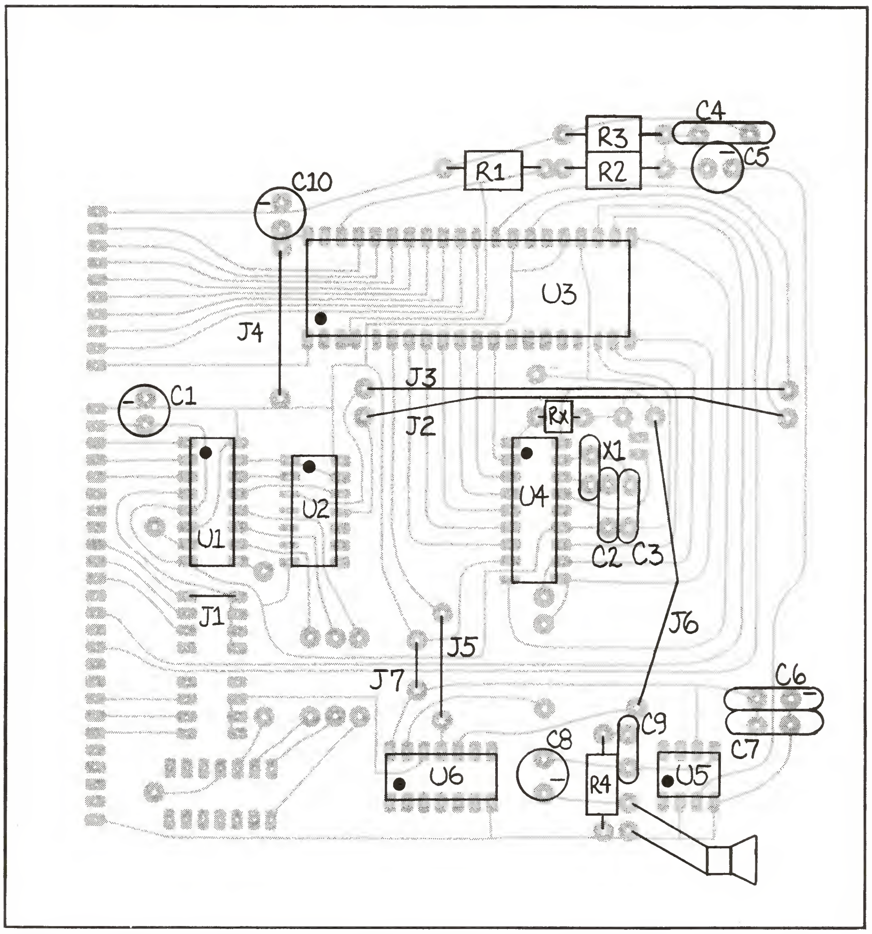


Figure 2: Layout of parts on the clock/sound board. Note dots in the corner of each chip (U1-U6) for correct orientation.

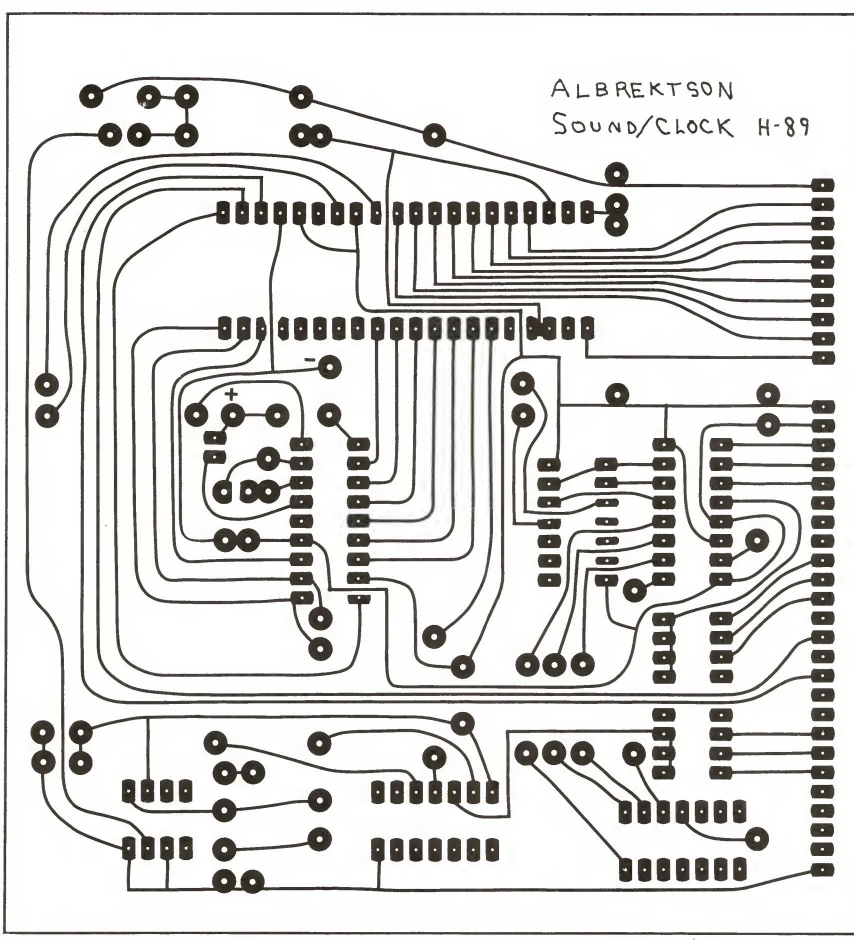


Figure 3: Camera-ready artwork for photographic production of the printed-circuit board.

ing them up—always touch the metal table first to eliminate any possible charge.

Begin construction by installing the 10-pin and 25-pin accessory connectors; the sockets for U1, U2, and U3; and jumpers J1, J2, J3, and J4. Using your ohmmeter, make sure that pins 1, 22, 23, and 24 on P–602 show infinite resistance compared with both pins 2 and 25. Also make sure that pins 2-9 on P–601 show infinite resistance to pins 1 and 10. These checks will ensure that the various power lines aren't shorted to ground, and can eliminate a lot of smoke from the "smoke test" when you finally turn it all on!

Carefully install U1 and U2. Make sure they are correctly oriented: pin 1 of U1 goes in pin 1 of the corresponding socket. (See Figure 2.) Be sure that no pins are bent under the IC, and that all pins go into their socket holes. Now the big moment is at hand. Slip the board onto a vacant accessory plug on the right side of the H89—and need I mention that power should be *off*?

Utter a quick prayer, and turn it on. The "H:" prompt should appear. Boot the system normally. Load MBASIC and run the program called STROBGEN.BAS (Listing 1).

Testing it with STROBGEN.BAS

This program will simply generate an I/O strobe at whatever address you want. With your logic probe touching pin 15 of U1, enter a "1" on the keyboard. This will send a strobe to I/O address 320Q and you should get a nice pulse on pin 15 of U1. Entering "2" should produce a pulse on pin 14, and "3" should do the same for pin 13. If you don't get these results, check the power and ground connections on U1 and be sure that the addressing is right. The jumper J1 should be connected to pin 9 of P–602—for addresses 320Q-327Q, that is! Be *sure* that nothing else in the computer is sharing those addresses!

If all is well, move to U2. Entering a "1" should produce a pulse on pins 3 and 11. Entering a "2" should only produce output on pin 3, and "3" should only produce a pulse on pin 11. If you don't get these results, check power and ground connections on U1, and be sure that U2 is indeed a 74LS00. (I spent a fruitless hour in the Twilight Zone before realizing that I had put the wrong chip in the socket for U2.)

Now that U1 and U2 check out, make a quick check of the empty PSG socket with the logic probe. Use the STROBGEN.BAS program and be sure that the pulses from U2 come through loud and clear on pins 27 and 29. At least the PSG will get the proper control signals if they do. Check that the 2.048 megahertz clock signal from the H89 is arriving at the PSG socket by looking for pulses on pin 2. Note that it is easy to accidentally touch pin 23 of the PSG socket—this

resets the PSG, and is connected to the H89 reset line. If you do touch pin 23, the H89 will probably reset, and you will need to reboot to continue checkout. Make sure you have pulsing on pins 30-37 (data bus), ground on pins 1 and 24, and +5 volts on pins 25, 28, and 40.

Sound generator construction and checkout

Turn off the computer, remove the board, and install U3 and U5 and all the resistors and capacitors associated with U5. To hook up an extension speaker to the sound section, I soldered a ⅛″ mini-jack to two wires which in turn were soldered to the clock/sound board at the pads indicated in Figure 2. Note capacitor polarity—the negative end of C5 goes to pin 2 of U5 and the negative end of C8 goes to the speaker connection. Naturally, the negative end of C6 goes to ground. The rest of the capacitors aren't polarized. Note: do not yet install the glitch-catching capacitors C1 and C10. That is for the very, very end.

All hooked up? Put the board in place, hook up your speaker, and turn it on. Check voltages on U3 and U5. Are they all getting the required 5 volts? If so, boot and load FIRSTEST.BAS (Listing 2).

Congratulations! You should now be hearing marvelous sounds from the speaker. If not, check to make sure the proper strobes are going to pins 3 and 11 of U2. Make sure data pulses are actually present on all of pins 30-37 of U3. Make sure clock pulses are present on pin 22 of U3. If all of this checks out and there is still no sound, double-check all of the audio circuitry surrounding U5. Is the polarity of the capacitors correct? Is U5 correctly oriented? Is the speaker hooked up? Is it a good speaker? If you get desperate, borrow an oscilloscope, run the above program, and see if you get an audio waveform on pins 3, 4, and 38 of U3. If yes, then everything but the audio amplifier is O.K. Redouble your efforts on the audio amplifier. If necessary, try a new LM386. I have encountered bad ones before.

Clock construction and checkout

Remove the board once again and install sockets for U4 and U5. Next, install jumpers J5, J6, J7, and also capacitors C2 and C3. Carefully install the MSM5832 chip. (Mind the static!) Install U6, and double check the #1 pin positions on U4 and U6. Install the board, turn on the computer, and make sure the PSG is still alive. (Run FIRSTEST.BAS.) If things still sound good, run the program called TESTOSC.BAS (Listing 3).

This program puts the clock into the "interrupt generating mode" so that pulses at the rate of one per second will appear on pin 10 of U4 and one per minute at pin 11. If the pulses are there and at the proper intervals, *hooray*, the clock is running! If not, first take your logic probe and check the logic levels on

```
1  '               STROBGEN.BAS
5  'THIS PROGRAM WILL GENERATE I/O STROBES AT 320Q-322Q
10 R = 208       'THIS IS THE DECIMAL EQUIVALENT OF 320Q
20 I = R+1       'IF YOU USED OTHER ADDRESSES, ADJUST ACCORDINGLY
30 D = R+2
40 T = VAL(INPUT$(1)) :ON T GOTO 50, 60, 70
50 OUT R,0:GOTO 40
60 OUT I,0:GOTO 40
70 OUT D,0:GOTO 40
```

Listing 1: STROBGEN.BAS will generate an I/O strobe at whatever address you want.

```
1    'FIRSTEST.BAS--INITIAL PROGRAM TO TEST PSG SOUND PART OF BOARD
5    DIM N(15)
10   R = 208            'DECIMAL ADDRESS TO SELECT PROPER PSG REGISTER
20   D = R+2            'DECIMAL ADDRESS TO WRITE DATA TO PSG
25                      'ADJUST LINES 10 AND 20 FOR OTHER THAN 320Q
30   FOR J = 0 TO 13    'READ DATA TO BE SENT TO PSG REGISTERS
40   DATA 155,0,77,0,58,0,0,248,16,16,16,107,39,10
50   READ N(J)
60   NEXT J
70   FOR J = 0 TO 13    'SEND 16 VALUES TO APPROPRIATE PSG REGISTERS
80   OUT R,J            'SELECT REGISTER J
90   OUT D,N(J)         'WRITE DATA N(J) TO REGISTER J
100  NEXT J             'DO AGAIN
110  STOP
```

Listing 2: FIRSTEST.BAS produces sounds for the initial checkout on construction of the programmable sound generator section of the clock/sound board.

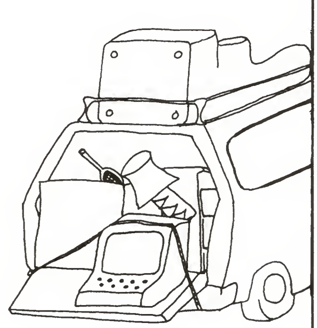
```
10 'TESTOSC.BAS--TEST PROGRAM TO PUT INTERRUPT SIGNALS ON CLOCK CHIP D0-D3
20 OUT 208,7      'SELECT MIXER REGISTER
30 OUT 210,128    'SET MIXER FOR OUT ON B, IN ON A
40 OUT 208,15     'SELECT PORT B REGISTER
50 OUT 210,252    'OUTPUT 252, WHICH PUTS A0-A3, CS & R HIGH; & HOLD LOW
60 PRINT"FAST PULSES SHOULD BE ON D0, 1 SEC ON D1, 1 MIN ON D2, ETC."
70 PRINT"IF NO PULSES ON D0, THEN OSC. IS NOT RUNNING."
```

Listing 3: TESTOSC.BAS is used after constructing the clock section. It puts the clock in interrupt-generating mode to produce regular pulses on pin 10 of U4.

```
10 'READTIME.BAS READ TIME/DATE/YEAR FROM PSG
20 DIM T(14), M$(12)
30 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY
35 DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
40 FORN=1TO7:READD$(N):NEXT N
41 FOR N=1TO12:READM$(N):NEXTN:N=0
50 R=208         'REGISTER SELECT PORT
60 D=210         'WRITE TO PSG PORT
70 I=209         'READ FROM PSG PORT
80 A=132+N*8     'CYCLE THROUGH EACH CLOCK COUNTER & COLLECT T(N)
90 OUT R,7
100 OUT D,128    'SET B FOR OUT, A FOR IN
110 OUT R,15
120 OUT D,A      'HOLD, WRITE = 0; CS = 1; A0-A3 AS DIRECTED
130 OUT R,14
140 T(N)=INP(I)-240
150 N = N + 1:IF N <> 13 THEN GOTO 80
160 S = T(0)+10*T(1)
170 M = T(2)+10*T(3)
181 IF T(5) = 0 THEN H=T(4):A$="AM":GOTO 200
182 IF T(5) = 2 THEN H=T(4)+10:A$="AM":GOTO 200
183 IF T(5) = 4 THEN H=T(4):A$="PM":GOTO 200
184 IF T(5) = 5 THEN H=T(5)+10:A$="PM"
200 W = T(6)+1            'TO BRING BACK TO SUNDAY=1 (INSTEAD OF 0)
210 D = T(7)+10*T(8)
220 MO = T(9)+10*T(10)
230 Y = T(11)+10*T(12)
231 Y$="19"+RIGHT$(STR$(Y),2)
240 PRINT H;":";M;":";S;A$;"  ";D$(W);"   ";M$(MO);D;Y$
250 N = 0:GOTO 50
```

Listing 4: READTIME.BAS will allow you to watch the clock tick off seconds.

Index to Advertisers



*MOVING? Don't lose your next issue of **Sextant** in the shuffle.*
Send your Change of Address to:
Sextant
Attn: Circulation Department
716 E Street, S.E.
Washington, DC 20003

*Contact advertiser directly.

pins 6 through 12 of U3 (the PSG). They should look like this:

PIN		
6	HIGH	— this is inverted to a low by U5 & allows clock to run
7	H	Setting all these high puts the clock chip into the "generate interrupt" mode
8	H	
9	H	
10	H	
11	H	Enables the READ function on the clock
12	LOW	Disables the WRITE function
13	don't care	

If the logic checks out, but no pulses are forthcoming on pins 9-11, then maybe the oscillator inside U4 isn't running. Check the values of C2 and C3. C3 should be a 20 picofarad unit, and a mylar or silver-mica one will always work. Some ceramic ones I tried didn't. Check to make sure that the outputs of the PSG Port B match the inputs to the clock chip: pin 18 low; pins 3-7 high, pin 2 low, and pin 8 high. If they don't match up, look for bad socket connections or a bad solder joint. If these are correct, and the clock still isn't running, either U4, U5, or the crystal may be bad, and it is more likely to be the clock chip than the crystal.

But that doesn't apply to you—you have nice, solid one-second and one-minute pulses! (There are one-hour pulses as well, but they aren't worth waiting for.) Instead, load and run the program READTIME.BAS (Listing 4) to watch the clock tick off seconds. To set the clock, load and run the program CLOCKSET.BAS (Listing 5). If you can correctly set all digits, your sound/clock/calendar board is practically done.

Finishing up

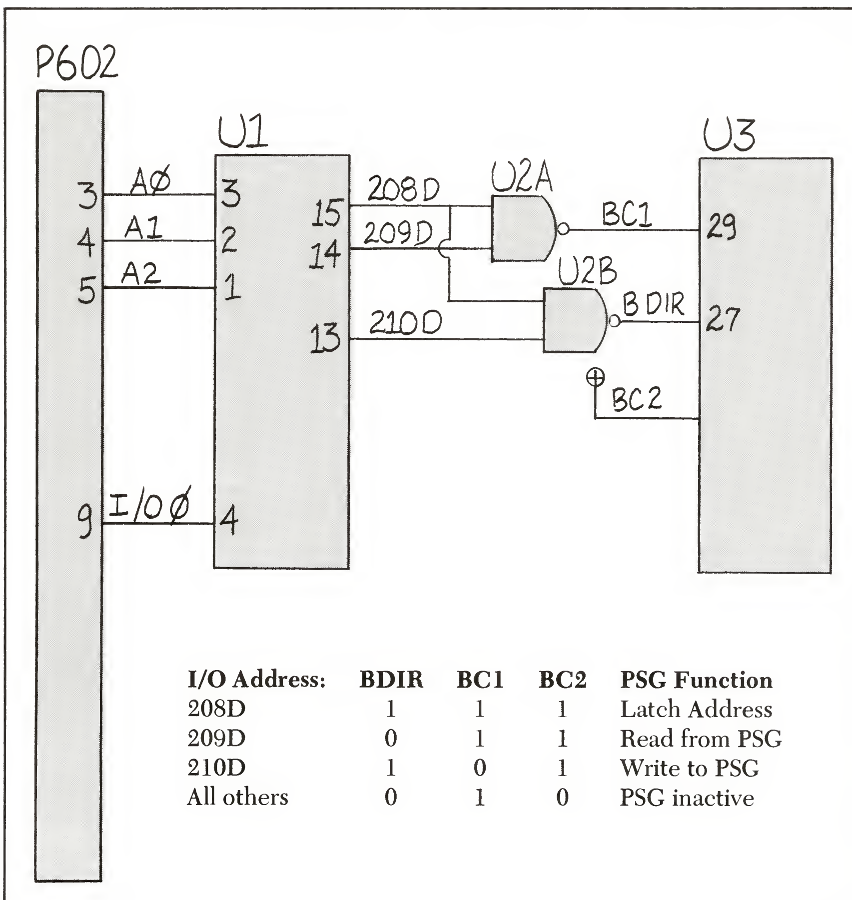
By now you may be wondering where the clock is getting its power. After all, the battery hasn't been hooked up yet! The answer is that the clock chip is getting its juice from the H89 via the chip select input (pin 9). To power the clock while the H89 is off, a simple battery pack is more practical than an elaborate rechargeable system. After all, it will run for years on a single 9-volt transistor-radio battery! Hook up the 9-volt battery to the board. Using a volt-

Listing 5: CLOCKTEST.BAS allows you to set the clock.

```
10 'PROGRAM TO SET MSM5832 CLOCK ON SOUND/CLOCK BOARD
20 DIM T(14), C$(14)
30 DATA SECONDS,MINUTES,HOURS,DAY OF WEEK,@
   DAY OF MONTH,MONTH,YEAR,AM/PM,ZERO SECONDS,ALL-AT-ONCE,DO AGAIN
40 FOR N = 0 TO 10 :READ D$(N):NEXT N
50 R=208          'PORT ADDRESS TO SELECT CORRECT PSG REGISTER
60 I=R+1          'PORT ADDRESS TO READ FROM SELECTED PSG REGISTER
70 D=R+2          'PORT ADDRESS TO WRITE TO SELECTED PSG REGISTER
80 INPUT"HOW MANY DAYS ARE IN FEBRUARY THIS YEAR? ";L
100 '***READ TIME FROM CLOCK INTO T(0)-T(12)
110 PRINT :N=0
120 A=132+N*8    'CYCLE THROUGH EACH COUNTER
130 OUT R,7
140 OUT D,128    'PORT B FOR OUT, A FOR IN
150 OUT R,15
160 OUT D,A      'SET ADDRESS & OTHER LINES
170 OUT R,14
180 T(N)=INP(I)-240
190 N=N+1:IFN=13THENGOTO210ELSEGOTO 120
200 '***DISPLAY TIME
210 PRINT "TIME: ";:IFT(5)=5ORT(5)=1THEN PRINTT(4)+10;:GOTO 230
220 PRINT T(4);
230 PRINT10*T(3)+T(2);10*T(1)+T(0);
240 IF T(5) = 5 OR T(5) = 4 THEN PRINT "PM";:GOTO260
250 PRINT "AM";
260 PRINT"   DAY OF WEEK:";T(6)+1;"DATE: ";
270 IF T(8) = 5 THEN PRINT 10+T(7);      'LEAP YEAR
280 IF T(8) = 4 THEN PRINT T(7);         'LEAP YEAR
290 IF T(8) < 4 THEN PRINT 10*T(8) + T(7);
300 PRINT10*T(10)+T(9);10*T(12)+T(11);
310 IF T(8) > 3 THEN PRINT "LEAP YEAR" ELSE PRINT
320 PRINT
400 '***DISPLAY TIME-SET OPTIONS & BRANCH TO SELECTED OPTION
410 FOR N = 1 TO 10:PRINT N,D$(N):NEXT N
430 INPUT"ENTER THE NUMBER OF THE UNIT TO CHANGE (OR <CR>): ";CH
440 ON CH GOSUB 460,500,540,570,640,680,720,760,2000,3000
445 CH = 0:PRINT:GOTO 110
450 PRINT:GOTO 110
460 '**CHANGE MINUTES
470 INPUT"NEW MINUTES: ";V
480 A=24:T=INT(V/10):GOSUB800
490 A=16:T=V-INT(V/10)*10:GOSUB 800:RETURN
500 '**CHANGE HOURS
510 INPUT"NEW HOURS: ";V
520 A=40:T=INT(V/10):GOSUB 800
530 A=32:T=V-INT(V/10)*10:GOSUB 800:RETURN
540 '**CHANGE DAY OF WEEK
550 INPUT"NEW DAY OF WEEK: (1-7)";V
560 A=48:T=V-1:GOSUB 800:RETURN
570 '**CHANGE DATE
580 INPUT"NEW DAY OF MONTH: ";V
590 A=64:T=INT(V/10):GOSUB 800
600 A=56:T=V-INT(V/10)*10:GOSUB 800
620 IF L = 29 THEN A=64: T=4+INT(V/10):GOSUB 800:RETURN
630 RETURN
640 '**CHANGE MONTH
650 INPUT"NEW MONTH: ";V
660 A=80:T=INT(V/10):GOSUB800
670 A=72:T=V-INT(V/10)*10:GOSUB800:RETURN
680 '**CHANGE YEAR
690 INPUT"NEW YEAR: ";V
700 A=96:T=INT(V/10):GOSUB800
710 A=88:T=V-INT(V/10)*10:GOSUB 800:RETURN
```

Please refer to the key to the right when completing the article evaluation form on the attached Reader Service Card above. By letting us know how interesting and/or helpful you found each feature in this issue of *Sextant*, you will help us choose the best articles for you in the future.

```
720 '**CHANGE AM/PM
730 A = 40
740 IF T(5)>3THENT=T(5)-4:GOSUB800:RETURN
750 IF T(5)<3THENT=T(5)+4:GOSUB800:RETURN
760 '**ZERO SECONDS
770 T=0
780 A=8:GOSUB800
790 A=0:GOSUB 800:RETURN
800 '***SUBROUTINE TO WRITE TO CLOCK
810 '     A = ADDRESS VALUES, T = DATA VALUES
820 OUT R,7
830 OUT D,192                      'SET B FOR OUT, A FOR OUT
840 OUT R,15
850 OUT D,0:FORJ=1TO10:NEXT J      'DELAY AFTER PUTTING IN HOLD
860 OUT R,14
870 OUT D,T                        'PUT DATA ON D0-D3 OF PORT A
880 OUT R,15
890 OUT D,A                        'PUT ADDRESS BITS ON PORT B
900 FOR J = 1 TO 10:NEXT J         'DELAY BEFORE WRITING
910 OUT R,15
920 OUT D,A+2                      'ADD WRITE COMMAND TO PORT B
930 OUT R,15
940 OUT D,128                      'TAKE OFF HOLD
950 RETURN
2000 '***SUBROUTINE TO ENTER MONTH, YEAR, DATE, DAY, HOURS,@
     MINUTES, SECONDS, AMPM
2010 INPUT"ENTER MONTH: ";MO1
2020 INPUT"ENTER DAY  : ";DA1
2030 INPUT"ENTER YEAR : ";YR1
2040 INPUT"ENTER DAY/WK:";WK1
2050 INPUT"ENTER HOURS: ";HR1
2060 V = MO1:GOSUB 660
2070 V = DA1:GOSUB 590
2080 V = YR1:GOSUB 700
2090 V = WK1:GOSUB 560
2100 V = HR1:GOSUB 520
2110 RETURN
3000 '***SUBROUTINE TO RE-DO M,Y,D,DAY,H DONE PREVIOUSLY
3010 GOTO 2060
```



I/O Address:	BDIR	BC1	BC2	PSG Function
208D	1	1	1	Latch Address
209D	0	1	1	Read from PSG
210D	1	0	1	Write to PSG
All others	0	1	0	PSG inactive

Figure 4: Generation of programmable-sound-generator signals. See text under "Programming the AY3-8910 PSG" for details.

meter, select a value for Rx that will result in about 4.5 volts at pin 1 of the clock chip. This has to be determined individually, since the MSM5832 varies considerably from unit to unit in its power consumption. All that remains is to install the glitch-catching capacitors C1 and C10 (mind the polarity!!) and wash off flux residue with trichlorethane.

What about left-side installation?

Once the board is up and running in a right-side slot, it can easily be adapted to run in a left-side slot. This involves isolating the clock board from all the signals that are present on the 25-pin connectors on the left side (except for power and ground), and "stealing" the signals it needs from a right-side board.

First, remove the 25-pin connector from the clock board and carefully cut off pins 3, 4, 5, 9, 10, 11, 13, and 15. This leaves only the power and ground connections intact. Re-install the 25-pin connector.

Next, decide which right-side board you want to use as a signal-stealer. I chose the serial board. (I have a horror of goofing up my disk controller!) Using a piece of six-conductor ribbon cable, connect pads 2, 4, 5, 9, 13, and 15 on the clock board to the corresponding pads on the serial board. If you used some other address than 320Q (the "middle" chip), then connect pads 10 or 11 instead of 9.

Programming the AY3-8910 PSG

In dealing with the PSG, I really prefer to program it directly. However, there is plenty of ready-to-run software available that takes practically all the effort out of generating music and sound effects. One excellent item is the Sound Device Driver, available for $10 from SigmaSoft and Systems of Dallas, Texas. Generating music and sound-effects is as easy as sending character strings to the sound device driver. Also, I offer a disk of sound and clock utilities for $10. This runs under the Heath Disk Operating System (HDOS) and includes a directory of other software sources supporting this sound generator and clock.

The AY3-8910 has great power, and programming it is not difficult to learn once you understand what the PSG is looking for. The PSG makes sounds depending on what values are stored in its 16 internal registers. In order for the H89 to read and write to these registers, the PSG responds to three commands: 1) latch register address from CPU into PSG (H89 tells PSG which register is to be read from or written to); 2) read from PSG (PSG transfers the value in the register indicated in the preceding step onto the data bus); and 3) write to PSG (the H89 transfers the value on the data bus into the PSG register indicated in Step 1).

See Figure 4 to discover how these commands are generated. Note how the

DEALERS DEALERS DEALERS

SSI WORD*PERFECT
—customized to the Z-100 running Z-DOS

Word processing with the clean screen look.

Function key technology provides most features at the press of a key.

* **ON SCREEN FORMATTING** provides a "what you see is what you get" document.
* **MATH, MERGE** packages, and **FOOTNOTE** ability included.
* **MACRO** function provides key stroke memory.
* **HELP** key provides on-line assistance with a keystroke.
* **PRINT BUFFER** supports two printers and will hold up to nine documents at one time.
* Supports most popular single or dual bin sheet feeders.
* **SUPERSCRIPT, SUBSCRIPT** and **HALF-LINE SPACING.**

Dual Document editing now supported.

HBM
Division of DP Service Inc.
5559 N. Elston Ave.
Chicago, IL 60630
(312) 774-2556

SSI Sorter and Speller now available.

Circle #181 on Reader Service Card

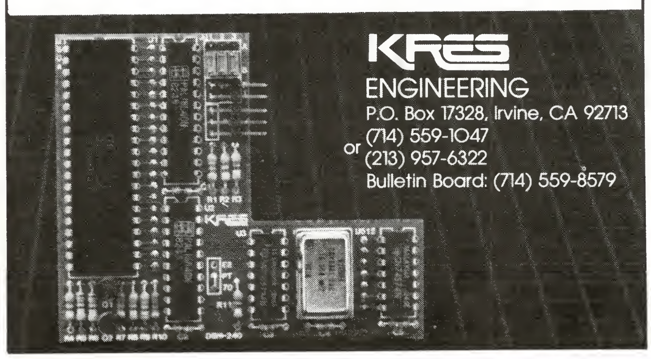
Double the Speed of your H/Z89-90
Cut computing time by as much as 50%

Fast Operation
With the Dual Speed Module, software that used to be slow and inefficient now zips right along. Run programs like SuperCalc (in "auto mode") and Spellstar without long waits.

Easy
Simply replace your existing microprocessor IC with the Dual Speed Module and stop wasting time. Installation requires no trace cutting or soldering.

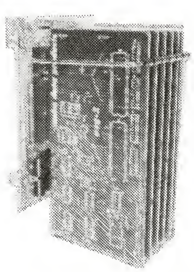
Fully Supported
Software support for CP/M and HDOS. Works with Magnolia Microsystems*, CDR, and many other disk systems.

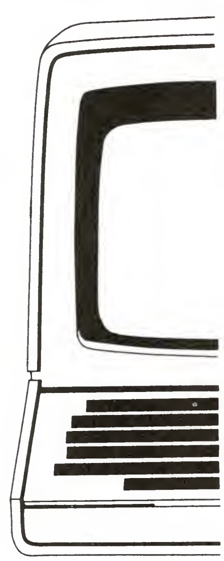
Dual Speed Module DSM-240 $79.95** + $3.50 shipping and handling (includes CP/M *or* HDOS; add $10.00 for both).
*Requires Software Patch PMM-100 $49.95 and Monitor ROM KMR-100 $59.95
**Prices valid through December 31, 1983
CP/M is a registered trademark of Digital Research



KRES
ENGINEERING
P.O. Box 17328, Irvine, CA 92713
(714) 559-1047
or
(213) 957-6322
Bulletin Board: (714) 559-8579

Circle #194 on Reader Service Card





H89/Z90's CAN NOW DEAL WITH A FULL DECK!

THE ORIGINAL ALL-IN-ONE ACCESSORY BUS EXPANDER.

MH89+3 doubles expansion capacity. Allows for for 6 right-hand type cards instead of the usual 3. Room at last to run those neat accessory boards you've seen advertised!

Piggyback motherboard installs internally with a screwdriver in just minutes – with no modifications! 3 slots exactly duplicate the originals. The 3 added slots occupy unused addresses and eliminate previous conflicts. 100% compatible with all accessory boards!

No overheating problems! Simple design draws little power. Leaves plenty of overhead for the minimal load of most accessories. Full technical information provided.

The best news about this "No-hassle" design is the price — **ONLY $150.** About 1/3 the price of other solutions!

Price includes assembled and tested MH89+3 expander, complete instructions and one (1) year warranty. CA residents add 6% tax. USA include $5 shipping. Foreign add $10. Telephone and COD orders accepted.

mako data products
1441-B N. RED GUM, ANAHEIM, CA 92806
PHONE (714) 632-8583

Circle #154 on Reader Service Card

THE SOFTWARE SUBSCRIPTION

VISED 3.0 is now available for ZDOS, CP/M-85, HDOS and CP/M-80. Now you can use the same editor in all four operating systems. VISED 3.0 is a visual editor with fast scrolling, searching and text manipulation. VISED 3.0 is versatile enough to be used as a text editor as well as a program editor. VISED/Z is sold as a two disk package for ZDOS and CP/M-85.

VISED is also distributed as **VIPROC** with TPROC, a text processor. TPROC is a text processor for applications requiring quality formatted text, such as letters, documentation, reports, and papers. It reads a file of text and commands and generates output to disk or printer. TPROC does filling, adjusting, centering, underlining and formatting with titles.

XREFBAS programs generate a variable name and line number cross reference list which can be output to the screen disk or printer. XREFBAS programs read from an ASCII file and display the line number being analyzed and all variable name or line numbers discovered on that line. The variable name and line number cross references may be generated together or the variable name cross reference may be generated separately. And best of all, they are FAST!

VIPROC/Z—VISED & TPROC (ZDOS & CP/M-85)	$75.00
VISED/Z—Visual Editor (ZDOS & CP/M-85)	$50.00
VIPROC/H—VISED & TPROC (HDOS)	$50.00
VISED/H—Visual Editor (HDOS)	$35.00
VIPROC/C—VISED & TPROC (CP/M-80)	$50.00
VISED/C—Visual Editor (CP/M-80)	$35.00
XREFBAS—Basic Cross Referencer (ZDOS/CP/M-85/80/HDOS)	$30.00
FTCOM—Remote Computer Communications (HDOS/CP/M)	$30.00

Visa/MasterCard 1 to 9 pm Pacific time 415-237-2115
Order Postpaid at PO Box 5379, Richmond, CA 94805
Calif. res. add 6% sales tax (BART counties 6.5%)

Circle #163 on Reader Service Card

Register		Function	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Decimal	Octal									
0	0	Channel A	8-bit fine tune A							
1	1	Tone period					4-bit coarse tune A			
2	2	Channel B	8-bit fine tune B							
3	3	Tone period					4-bit coarse tune B			
4	4	Channel C	8-bit fine tune C							
5	5	Tone period					4-bit coarse tune C			
6	6	Noise period				5-bit period control				
7	7	$\overline{\text{Enable}}$	$\overline{\text{In}}$/Out		$\overline{\text{Noise}}$			$\overline{\text{Tone}}$		
			IOB	IOA	C	B	A	C	B	A
8	10	A amplitude				Mode	4-bit amplitude			
9	11	B amplitude				Mode	4-bit amplitude			
10	12	C amplitude				Mode	4-bit amplitude			
11	13	Envelope	8-bit fine tune envelope period							
12	14	Period	8-bit coarse tune envelope period							
13	15	Env. shape/cycle					Cont.	Att.	Alt.	Hold
14	16	I/O port A	8-bit parallel I/O port A							
15	17	I/O port B	8-bit parallel I/O port B							

Figure 5: The 16 internal registers of the programmable sound generator and the functions they control.

three I/O addresses, 320Q-322Q, are decoded into control signals for the PSG. Note that U1 has access to bits A0-A2 of the H89 address bus. It responds to the I/O strobe on pin 9 and decodes it based on the address on A0-A2. This produces strobes on pins 7 and 9-15 that correspond with the I/O address. U2 combines these into two control signals that control the three things that the PSG can do: read from PSG, write to PSG, and latch address into the PSG. When an I/O command addressed at 320Q is issued, pin 15 goes low. This causes the outputs of U2A and U2B to go high, which the PSG interprets as "Latch address into PSG."

Without going into too much detail, the number of the PSG memory register to be written to or read from is now on the data bus; and when the PSG gets this command, it latches it into its internal memory so that at the next READ or WRITE instruction, the PSG knows which of its 16 registers is being referenced. An I/O command addressed to 321Q causes pin 14 of U1 to go low, causing only pin 3 of U2 to go high. This is interpreted by the PSG as "Read from PSG." This causes the data in one of the 16 PSG registers to be transferred onto the data bus and to be read by the H89. The last option, an I/O command at 322Q, therefore brings pin 13 of U1 low and pin 11 of U2 high, and means "Write to PSG." The data on the data bus is then written into the PSG register indicated by a previous "Latch register address" I/O instruction.

The 16 internal registers of the PSG control its many functions (Figure 5). Register pairs 0-1, 2-3, and 4-5 are all tone control registers; the frequency of each of three separate oscillators can be individually set by writing the correct values into these registers. (For detailed information on which values to use to produce a particular frequency, consult the AY3-8910 Data Manual, available from parts suppliers.)

Register 6 is the "noise period" register; the noise generator can be made to sound high and sharp, or low and boomy. Different explosion noises can be produced by selecting different noise periods.

Register 7 is the most important to understand thoroughly. It is the "mixer register" and turns on any of the tone and noise effects. It is also used to set the directionality of the two I/O ports on the PSG, called Port A and Port B. In the section on communicating with the clock chip, we will see how these ports work.

Registers 8, 9, and 10 are the volume control registers for the three tone channels. Volume levels from 0 to 15 can be written into these registers to control volume. If the volume is set at 16, the actual volume will track the envelope determined by the envelope generator, which is controlled by registers 11, 12, and 13.

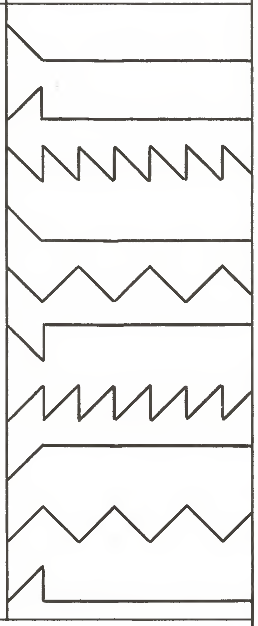
Registers 11 and 12 together set the period of the envelope. The envelope period can range from a short fraction of a second to almost nine seconds. The envelope shape has eight options, each selected by writing the appropriate bits in register 13. (See Figure 6.)

How the clock chip is hooked in

Remember that one of the freebies with the AY3-8910 sound chip is the availability of two 8-bit parallel ports! One of the ports (Port "A") will be used to transfer data to and from the clock chip; and the other port (Port "B") will be used to control the clock chip. Port A uses four lines to interface with the clock chip's four data lines (D0-D3). Seven lines of Port B are used to control the clock's READ, WRITE, and HOLD inputs

R13 Bits B3	B2	B1	B0
0	0	X	X
0	1	X	X
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Graphic representation of envelope generator output.



B3: Continue
B2: Attack
B1: Alternate
B0: Hold

Figure 6: The envelope shape has eight options, each selected by writing the appropriate bits in register 13.

and to provide address information A0-A3 to select the clock chip's internal counters.

Take a look at Figure 7. Notice that the clock chip has 13 internal registers, and that they may be either read from or written to. The selection of the correct register depends on the value of the address inputs A0-A3. (Note that these address lines have nothing to do with the H89's address lines!)

For example, in order to read the M10 counter, the four-bit address the counter needs would be 1100B (binary). Our first step would be to program Port B for output, and then put 1100B onto B3-B6 (bits 3 through 6 of Port B). To program Port B for output, we use register 7 of the PSG. The value to be written into that register to produce output on Port B and input on Port A would be 10XXXXXXB, where the X's indicate "Don't care." From an MBASIC program, we could use OUT 208,7 to select register 7 of the PSG, followed by OUT 210,128 (where 128D=10000000B), thus setting the PSG for output on B and input on A.

Reading the clock chip

The next step is to figure out what to put on the Port B lines that control the clock chip that would result in reading the M10 register. Bit 0 could be anything, since it remains unconnected. Bit 1 should be 0 since we want to read, not write; and bit 2 should be 1 for the same reason! Bits 3 through 6 control the

clock chip's register addressing, so they would be B3=0, B4=0, B5=1, B6=1. With these values, the A0-A3 lines on the clock chip would see 0011B, which corresponds to the M10 register.

Finally, B7, which governs the clock's HOLD line, could be either low or high. When low, it sets the HOLD line of the clock high, and enables accurate reading and writing. In practice, it is not necessary to HOLD the clock for reading, but it is required when writing. In this particular example, we will not HOLD the clock, so B7 should be high.

Putting this all together, we find that the eight bits to appear on Port B should be 1001110XB. The decimal equivalent of 10011100B is 156D. If we put 156D on Port B, the clock chip will be more than happy to put the M10 data on its D0-D3 lines. To put 156D on Port B, we do an OUT 208,15 to select Port B, then an OUT 210,156 to actually write the data.

Hang in there—we're almost done. All we have to do is program the PSG to read Port A, and transfer the information to the H89.

First, indicate that Port B is to be addressed by doing an OUT 208,14 followed by M10=INP(209) and at last we have our desired information. If it is 2:28 a.m., then we should have M10=2, right? Surprise! Upon inspection it is 240D too high! What is going on? Note that not all eight lines of Port A on the PSG are used. The high four bits are left floating, and internal pull-up resistors make the high-order bits all ones! In reading from Port A the PSG always sees "1111XXXXB." So you must always subtract that 11110000B (240D) from the values read from the PSG.

Writing to the clock chip

Writing to the clock chip only has to be done once—to set the time! It is a bit tougher than reading since timing is more critical. Here is the drill; first set both ports for output. The Register 7 value for this is 11XXXXXXB. Issue the commands OUT 208,7 and OUT 210,192 (192D=11000000B). Next put a HOLD on the clock. To do this, just make B7 low: OUT 208,15 followed by OUT 210,0. That will bring everything on Port B low; and as the low on B7 is inverted by U6A, it will cause activity in the clock registers to crash to a halt.

Next, the data to be written has to be put on Port A. If you want to write a 4 into the M10 register, then start by putting "4" on Port A: OUT 208,14 and OUT 210,4 should do the trick. The address of the proper clock counter needs to be put on Port B next. Since the data on Port B is already 00000000B, all we have to do is calculate the address value for the desired clock counter and put it out Port B—the clock will continue to HOLD since B7 will still be 0. Since the M10 register address is 1100, we want to put 00011000B on Port B. This is 24D, so do an OUT 208,15 and OUT 210,24. We now have register M10 selected, a HOLD in effect, and the data to be written all in place on Port A.

All that remains is to switch the WRITE bit (B1) high for a fraction of a second. To turn on the B1 bit, just add 2 to the value already on Port B. The commands

Register address	Register contents	Port B Value: To read registers: (includes READ bit) (HOLD bit is off)	Port B Value: To write registers: (Doesn't include WRITE bit) (HOLD bit is on)
0	S1	132	2
1	S10	140	10
2	M1	148	18
3	M10	156	26
4	H1	164	34
5	H10	172	42
6	W	180	50
7	D1	188	58
8	D10	196	66
9	MO1	204	74
10	MO10	212	82
11	Y1	220	90
12	Y10	228	98

Table 2: This table indicates which values are to be put on Port B in order to read from or write to a desired clock register. For example, if you want to write to the Y1 register, just put 220 (decimal) on Port B.

A0	A1	A2	A3	Internal Counter	D0	D1	D2	D3	Data Limits
0	0	0	0	S1 note (1)	*	*	*	*	0–9
1	0	0	0	S10	*	*	*		0–5
0	1	0	0	M1	*	*	*	*	0–9
1	1	0	0	M10	*	*	*		0–5
0	0	1	0	H1	*	*	*	*	0–9
1	0	1	0	H10	*	*	(2)	(3)	0–1 or 2
0	1	1	0	W	*	*	*		0–6
1	1	1	0	D1	*	*	*	*	0–9
0	0	0	1	D10	*	*	(4)		0–3
1	0	0	1	MO1	*	*	*	*	0–9
0	1	0	1	MO10	*				0–1
1	1	0	1	Y1	*	*	*	*	0–9
0	0	1	1	Y10	*	*	*	*	0–9

Notes:
*Indicates that data is valid as 0 or 1. Blanks appear as 0 during a READ and are unrecognized during a WRITE.
1. Writing anything to the S1 or S10 zeros both regardless of D0-D3.
2. D2 is 1 for p.m. and 0 for a.m. when in 12-hour mode.
3. Writing 1 to D3 sets it to 24-hr mode. Writing 0 to D3 puts it in 12-hour mode.
4. Set D2 for 1 if leap year, otherwise set D2 at 0.

Figure 7: Internal registers of the clock chip.

```
1  'WRITDEMO.BAS--WRITES DATA IN LINE 20 INTO CLOCK
5  DIM D(13)
10 FOR N = 0 TO 12 :READ D(N): NEXT N
20 DATA 1,1,2,2,3,0,4,2,2,1,1,9,9 'NOV22-99,WED,3:22:11AM
60 OUT 208,7               'SELECT MIXER REGISTER
70 OUT 210,192             'SET PORTS A AND B FOR OUTPUT
80 FOR N = 0 TO 12
100                        'WRITE D(N) TO THE CLOCK
110 OUT 208,15             'SELECT PORT B
120 OUT 210,0              'PUT ON HOLD BY MAKING B7 = 0
130 FOR N=1TO5:NEXT N      'WAIT A BIT FOR COUNTERS TO SETTLE
140 OUT 208,14             'SELECT PORT A
150 OUT 210,D(N)           'PUT DATA VALUE D(N) ONTO PORT A
160 OUT 208,15             'SELECT PORT B
170 A = 2 + 8*N            'COMPUTE COUNTER ADDRESS FROM VALUE OF N
180 OUT 210,A              'PUT ADDRESS VALUE ONTO B PORT KEEPING HOLD ON
190 FOR N=1TO5:NEXT N      'WAIT A BIT FOR ADDDRESS TO SETTLE
200 OUT 208,15             'SELECT PORT B
210 OUT 210,A+2            'TURN ON WRITE--TRANSFER DATA INTO CLOCK
220 OUT 208,15             'SELECT PORT B
230 OUT 210,128            'TURN OFF HOLD BY MAKING B7 = 1
240 NEXT N
```

Listing 6: WRITDEMO.BAS is a program to write to the clock registers.

```
5  'READDEMO.BAS--READS CLOCK COUNTERS INTO D(0)-D(12)
10 DIM D(13)
40 OUT 208,7               'SELECT MIXER REGISTER
50 OUT 210,128             'SET PORT B FOR OUT, PORT A FOR IN
60 FOR N = 0 TO 12         'READ ALL 13 CLOCK COUNTERS
70 A = 132 + N*8           'CALCULATE ADDRESS VALUES
80                         'NOTE THAT A 4 FOR THE READ BIT IS INCLUDED
90 OUT 208,15              'SELECT PORT B
100 OUT 210,A              'PUT ADDRESS & READ ON THE B REGISTER
110 OUT 208,14             'SELECT PORT A
120 D(N)=INP(209)          'READ DATA FROM PORT A
130 D(N)=D(N)-240          'CORRECT FOR D4-D7 BEING HIGH
140 NEXT N
```

Listing 7: READDEMO.BAS is a program to read from the clock registers.

OUT 208,15 and OUT 210,26 would do the job. The data on Port A gets transferred to counter M10 in the clock, and as soon as the HOLD is removed, activities will commence on schedule. Note that as long as the HOLD is on for less than a second, the internal registers will keep their correct time. We take the HOLD off by writing something like 1XXXXXX to Port B: OUT 208,15 and OUT 210,128 (1000000B=128D).

The reason for the inverter in the HOLD line is related to the presence of internal pull-up resistors in the PSG. The PSG is RESET when the computer is turned on, and both the PSG ports go to 11111111B at RESET. If the HOLD input of the clock was directly connected to one of these ports, it would go into immediate HOLD, and the time would suffer as a result. These same resistors in the PSG serve the clock chip, which would otherwise require external pull-ups.

If all of this register-juggling seems hardly worth the effort, and you're thinking you'll spring a few hundred bucks for that Hayes Chronograph after all, wait! All you *really* have to do is use the handy-dandy table (Table 2) that tells you exactly what to put on Port B in order to read or write a desired clock register. For example, if you want to write to the Y1 register, just put 228D on Port B. To write the D1 register, put 58 on Port B. A couple of sample MBASIC programs that do just this are WRITDEMO.BAS and READDEMO.BAS, contained in Listings 6 and 7.

A few last words . . . .

Once you have the parts on hand, assembling and testing the clock/sound board is just a Saturday afternoon project, and the results are worth it! You have the time and date available at all times, callable from BASIC or assembly-language programs. The sound effects that are possible are just amazing! The blend of sirens, flying saucer noises, buzzes, and explosions were just what my kids were yearning for, and my once-scorned "educational" games are now the season's biggest hit!

Ordering Information

Printed-circuit board, $30, postpaid
Sound and clock utilities, $10, postpaid —HDOS 5" hard-sectored disk format only
Ray Albrektson
A.C.P.O. Box 51
Quezon City 30001
Philippine Republic

Sound Device Driver, $10
Sigmasoft and Systems
4488 Spring Valley, #107
Dallas, TX 75234
214/392-1025

Integrated-circuit chips:

U1	74LS138	8-line decoder
U2	74LS00	Quad 2-IN NAND gate
U3	AY3-8910	PSG (Note 1)
U4	MSM-5832	Clock chip (Note 2)
U5	LM386	Sound amplifier
U6	74LS04	Hex inverter

Connectors:

Sockets:

for U1	16-pin
for U2, U6	14-pin
for U3	40-pin
for U4	18-pin
for U5	8-pin

Right-angle headers: (Note 3)

P601 10-pin	Molex part # 22–15–2101 Heath part # 432–1074
P602 25-pin	Molex part # 22–15–2251 Heath part # 432–1076

Resistors, capacitors, diodes, misc.:

X1	32.768 crystal (Note 2)	
B1	9 volt	battery clip
C1, C10	5 microfarad	capacitors
C2	5–30 picofarad trimmer	capacitor
C3	20 picofarad	capacitor (mica)
C4	300 picofarad	capacitor
C5	2 microfarad	capacitor
C6	10 microfarad	capacitor
C7	0.1 microfarad	capacitor
C8	250 microfarad	capacitor
C9	0.05 microfarad	capacitor
R1	1,000 ohm	resistor
R2	5,000 ohm	resistor
R3	500 ohm	resistor
R4	10 ohm	resistor
Rx	varies with U4	resistor

Note 1: available from:
Digital Research Computers
P.O. Box 40156
Garland, TX 75040
214/271-3538
MasterCard and Visa accepted

Note 2: available from:
Concord Computer Products
1971 South State College
Anaheim, CA 92806
714/937-0637
MasterCard and Visa accepted

Note 3: available from:
Force Electronics
343 Hindry Avenue
Inglewood, CA 90306

Note 4: an 8-ohm speaker is also necessary.

Etched & Drilled Printed Circuit Board available from author for $30 postpaid. Send check or money-order to:
Ray Albrektson
A.C.P.O. Box 51
Quezon City 3001
Republic of the Philippines

Software available from the author, on 5" hard-sectored HDOS disk only, includes SOUNDED.BAS (sound-effects editor), CK.DVD (clock-setting utility), prologue for setting system date on boot-up, and information on including time information in directory files, etc. Price is $10 postpaid.

Table 3: Parts list. Except as noted, most parts should be available from local suppliers. (Also see ordering information at the end of the article.)

Circle #161 on Reader Service Card

Circle #235 on Reader Service Card

Circle #230 on Reader Service Card

Circle #108 on Reader Service Card

How One Family Learned to Live with an H89: A Dialogue

Pioneering on the frontiers of technological change is not without its costs.

Skip and Pam Chambers

Skip: I am writing this history of my affair with a Heathkit H89 computer as a warning to other users and prospective purchasers. While organizing my thoughts, my wife felt that some of her comments would also be appropriate. Therefore, what follows is our recollection of how a happy family learned to live with an H89.

Pam: *First of all, I'm not even using a word processor to write this. I avoid any direct contact with our home computer for fear of having to use it.*

Skip: It all began in the spring of 1980. The personal computer was still a fairly new idea. People were just becoming aware of them. Computer stores were opening, magazines were discovering them, but nobody seemed to know much about them. I saw an ad for a National Technical Schools correspondence course on computers. Part of the course was building your own Heathkit computer. It was an H89 with 16K RAM (16 kilobytes of random-access memory—it sounds bigger when you say it that way, doesn't it?).

Having built a Heathkit digital multimeter, this looked like the way for me to enter the world of computers. After all, how much harder could a computer kit be?

By Thanksgiving, I had received the first part of the computer: the bottom. Assembling the base was no big deal. After all, I still had my digital multimeter to check my work. The base was all I had, though. It was March before the next kits came. Since my family was skeptical about the whole project, I was quick to point out how useful the base had already been as a doorstop and paperweight. They didn't seem impressed.

Eventually, the basic machine was completed. One entire Sunday was spent soldering little plugs on little wires for the cassette interface. Everyone had gathered around for the powering on ceremony. As far as I was concerned, it was great! My three-year-old's only comment was "Why doesn't it talk?" In retrospect, perhaps it would have been better to stop right there. (Does anyone need a slightly used cassette interface?)

Build a computer

The disk controller board and drive were installed later in the week. It was at this point that my education really began.

Lesson 1: When the book says to remove a circuit board before changing parts, do it.

Lesson 2: Don't drink more than two or three cans of beer before plugging integrated-circuit (IC) chips into a computer.

Lesson 3: ICs are really tough little gadgets. Installed wrong, they can really get hot.

The next day, after parts and tempers had cooled, we had a working H89 computer. Our family had entered the computer age. Our budget and records could now be properly organized. The kids would be computer literate. With all the time we would save, life would be wonderful. (Imagine a happy family gathered around the computer, all smiling with music in the background, just like in the TV ads.)

> ***I'm scared that someone will swallow an integrated circuit with the mashed potatoes.***

On page 2 of the manual for the Heath Disk Operating System (HDOS), it says that 32K of memory is required to run HDOS. Call Heath and wait for parts. Page 3 says "Insert a blank diskette...." Call Heath and wait for parts. United Parcel Service is really quite efficient. The first boot only took slightly over a week.

The system ran just fine; budget data was entered—after buying the PIE editor. Things worked as advertised—after buying Microsoft BASIC and lots of diskettes—but not as expected.

Skip left a set of instructions with me so that the children can use the computer when he is away. The instructions begin with "plug in" and end with his office phone number. I live in fear that one of the kids will say, "Change the disk, Mommy."

The kids did not really want to be computer literate, I wasn't saving any time, and none of us really knew what to do with the thing anyhow. After all, how often can you do your budget each month?

A couple of years ago, when Skip was trying to convince me of the value of having a home computer, he interested me by saying that all my recipes could be organized, filed, and stored ready at my fingertips. I have thousands of recipes stuffed in drawers, languishing as bookmarks, and being used as scratch paper. Dreams of an organized kitchen and a printout as a shopping list filled my head. Well, that was two years ago and my recipe disk contains a single recipe for strawberry meringue pie, which I never make because I don't know how to put the disk in.

Rebuild SYSCMD.SYS

It was then that I discovered HUG—the Heath Users' Group. While browsing at my local Heathkit store, I found a collection of *REMark* issues. These became the source of many projects.

One article contained a program providing enhancements to the HDOS command, STAT, by which you find out the status of your various devices. The program worked just fine, but I felt it really ought to be included in HDOS's command processor, SYSCMD.

So I purchased the Heath course on assembly language and went to work on the problem. In just three months of non-stop work, I had picked SYSCMD apart, learned how it worked, and put it back together with the new STATus code included. Of course, a few problems arose along the way. I deleted my source-code file by mistake, crashed disks, and became frustrated with some of HDOS's limitations.

This led to more enhancements. SYSCMD got an UNDELETE command, clock support, abbreviated commands, and more flexibility. While this was very nice, it took all summer. (My four-year-old asked if the computer could talk yet. It couldn't.)

Since I bought the computer to save time, I had to make it run faster. So I looked around for areas where improvement was obviously called for. Well, obvious to me, anyway. One that I saw was PIP, the peripheral interchange program. It worked well enough at its job,

Circle #183 on Reader Service Card

Circle #113 on Reader Service Card

handling things like COPYing and getting DIRectory lists. But every time something like that would be done, SYSCMD would have to process my command and then load PIP to do the job.

So one obvious area of improvement was to eliminate PIP and add its function to SYSCMD. This would save loads of time not doing all that work to load PIP every time I wanted a directory list.

There was only one problem: it wouldn't work.

By Halloween, though, the end was in sight. I missed the trick-or-treating to work on it. Thanksgiving was a blur. For some reason, my H89 was moved from the house to the garage the day before Christmas. New Year's Day came, and the program worked! PIP was gone. Pam seemed less than thrilled with this news. In fact, she wasn't talking to me much at all.

Finally, I had the time to enter the budget data for the last three months. I was only a little further behind than before we had a computer to help.

My five-year-old asked if I would now make the computer talk. I decided that was a good idea, as soon as I installed the accelerator, the HUG modification to make the computer run twice as fast.

It was here that I forgot Lesson 2, and by now the beer consumption was away up. Did you know that the window on an EPROM chip can glow like a little lamp? It is really an impressive sight. The accelerator had to work because I needed the speed for my next project: rewriting HDOS. It did, too, in only four tries.

Then there are those fascinating little projects and modifications that consume him for months on end. He walks in the door from work and turns on the computer before he says hello. He jumps out of bed in the morning and lets the computer "warm up" while he shaves. Seems like a great method of birth control, rather expensive though. I sometimes hold the children up in front of the screen and say "This is Mandy. She is five now."

Those "little modifications" look like major construction projects to me. We are a regular stop for the local UPS truck. When Skip is trying to "make a few changes" to something, I know I'm going to lose my kitchen counter space to integrated circuits and soldering irons. I'm scared that someone will swallow an integrated circuit with the mashed potatoes.

By this time, I understood HDOS well enough to know that it used too much memory and too much disk space. Following my now proven, if inefficient, methods, I managed in just three months to rewrite HDOS and its overlays.

Since I had put the work into what was now really a new product, I decided to offer it for sale. I sent a letter with a product announcement to *Buss* newsletter and they carried it. I had become an independent supplier to the Heath/Zenith community.

My version of HDOS, which is organized completely differently from Heath's, uses only one-half the disk and memory space, and has more features. Finally my budget and MBASIC will fit into the computer.

Reflect upon one's successes

There is my organized, efficient husband who puts all our household budget and records on the computer. We used to have a four-drawer file cabinet with lots of folders clearly labelled: Life Insurance; Taxes; etc. Now everything in our life is on one tiny disk. If he kicks the bucket before I learn what to do when the display says BOOT, I'll never be able to collect the insurance.

After almost three years, we have a running H89 with HDOS and an automated budget. This has been a worthwhile and fascinating experience. For example, from an analysis of my budget I have learned that I really can't afford a personal computer. My children may not be computer literate, but they have learned many new words.

I really resent our computer's record-keeping ability. All of our receipts are immediately entered so at the end of each month he can see how much I went over budget. In the "good old days," we had a shoe box full of receipts and I could slip a purchase or two right by him. Sometimes he didn't even bother to add up all of those little numbers, but not now. With just a couple of furious swipes at the keyboard, I can get yelled at in record time.

Oh yes, I finally bought a Type-N-Talk speech synthesizer from Heath. When I showed it to my six-year-old, she didn't seem too interested. Kids just don't have any patience today.

Our computer has a voice and I am the only one in the family who can't understand it. I ask my six-year-old to translate.

I have just taken inventory of my programs. While no single application, alone, can justify the cost of the computer, taken together they might.

For example, it is a super word processor. At the present time that is no big deal, since I only process a few words a year. It won't be long until the kids are using it for term papers, though. I wrote a nifty subroutine to do my income tax return as part of my budget program. (It reduced my taxes so much that my return is being audited.)

One of our other high tech purchases is a video tape machine. Kids like cartoons, and cartoons are short so you can get a lot of them on a hundred or so tapes. Kids also like to watch a specific cartoon when they want it. The computer is the only practical way to index and locate titles in an ever changing cartoon population.

The list goes on.

And reflect a little bit more?

If you detect a tone of fear on my part, you may be right. It may be hard for the majority of you to believe, but there are mechanical morons out here. We're the type whose palms get sweaty just looking for the on-off control on a tuner (which used to be called a radio). When the car's temperature light glows red, we're the ones who turn on the air-conditioning to cool things down. We even have trouble using a plumber's helper effectively. With this mental make-up, the computer revolution is just about as comforting to us as the French Revolution was to Marie Antoinette.

Actually, it is an awesome revolution. One that has a potential of changing nearly every aspect of our lives. And, yes, that can be frightening!

Having a hacker for a husband has forced me to think about things. If we're concerned about the quality of the relationship between husbands, wives, children, and friends, will the addition of the computer in so much of our lives be a benefit? Or will we be more isolated from each other because of it? Or, if I put aside my suspicions and jump right in, will it free me from a lot of the drudgery of a 9 to 5 existence?

It is exciting to see six- and seven-year-old children practicing math skills using the computer. They get a kind of one-on-one experience the teacher can't give them. But, then again, this one-on-one experience is with a machine. I've seen a child's face light up as bright as the red and green SUPER spelled out on a screen after he has completed a series of problems. Then I've seen him look around for someone to see his triumph.

Can you see where we are heading? There's no harm in being patient with those of us who have our doubts. We'll join you in this revolution in spite of them.

Just being a computer "hacker" is not all bad either. I believe my HDOS/SYSCMD is one of the most convenient products around. Some people work crossword puzzles or play chess for recreation. I like to find new ways to do things on my computer. It can be an interesting, if frustrating, mental challenge.

There is just one thing in the back of my mind that bothers me as I spend my time with the H89. Do you remember all those jokes and science-fiction stories about building The Ultimate Computer? There's a one liner that I recall every so often.

This company built the ultimate computer. The first question they asked it was, "Is there a god?"

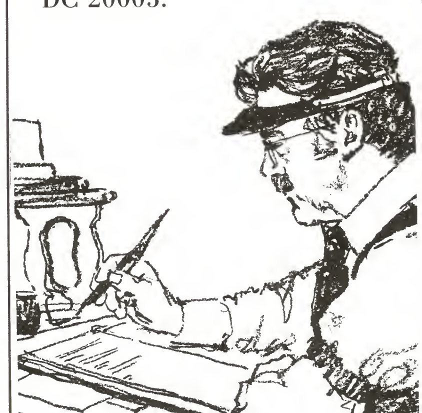
It replied, "There is now!"

Ordering Information

One Module HDOS and New SYSCMD:
$30 each or both for $50
Skip Chambers
3822 Westminster Drive
Carrollton, TX 75007

Authors:

Have you considered writing an article for *Sextant*? We're looking for human-interest stories, in-depth technical features, program descriptions, product reviews, and tutorials which relate specifically to Heath/Zenith systems. We pay more than many other microcomputing magazines. Send your article, idea, rough draft, or outline to: John Walker, Technical Editor, Sextant, 716 E Street S.E., Washington, DC 20003.



Classified Ads

To get in touch with your fellow Heath/Zenith users, place your short notice in *Sextant*'s classified section. The rate is only 75¢ per word with a minimum of 15 words. Send your typewritten ad and payment to: *Sextant*, Classified Ad Department, 716 E Street S.E., Washington, DC 20003. (Please include your name and telephone number for our records.) We need to receive ads for the May–June 1984 issue by March 9, and for the July–August issue by May 11. If you have any questions, contact Sharon Conaway at 202/544–2868.

Luminaries

Ray Albrektson is currently an extension faculty member at the International School of Theology—Asia, in the Republic of the Philippines. He's looking into the possible Third World uses of computers in education. (With and without sound effects, presumably.)

Skip and Pam Chambers both attended Purdue University (M.S., Mechanical Engineering, and B.S., Psychology, respectively). He now works in the data processing industry in communications. She, contrary to the impression you might have, is helping their local Texas school system implement its computer-literacy effort. (Unfortunately, she has to deal with Another Brand at school. Might that explain something about her attitude toward computers?)

Charles Cohn is a physicist at the Argonne National Laboratory near Chicago. He works in computerized data acquisition and control systems for experimental nuclear reactors. (We still don't know how much of that work is done on a single-drive '89.)

Leonard E. Geisler got into electronics in the '30s, and into computers as a publications engineer in the '60s. Perhaps his interest in forms came later: he's Secretary Pro Tem of the A-SQR (Ann Arbor to outsiders) Heath Users' Group, in Michigan.

Dean Gibson has his bachelor's degree in Mathematics from the California Institute of Technology, and his master's in Computer Science from the University of Wisconsin. He's best known to most of us through his consulting company, UltiMeth, and his Heath/Zenith hardware support code. But besides ordinary folks, his customers have also included big guys like the Air Force and National Semiconductor. It pays to be organized.

Siebert Ickler is Systems Manager for the H.E.B. Grocery Company, headquartered in Corpus Christi, Texas. The job involves both micros (Heath/Zenith among them) and an IBM mainframe. So a little thing like an ESCape # won't slow him down, even if he has to slow *it* down.

Terry Jensen is the National Heath Users' Group's software developer. He's also system operator for the HUG Bulletin Board on CompuServe's MicroNet. So he gets to keep an eye on what folks are doing around the country.

Charles W. Rogers is a free-lance computer scientist. He divides his time among teaching, consulting, and farming in the southern Missouri Ozarks. He started out in the computer field writing assembly-language input/output drivers in the days of vacuum tubes. A programmer was a lot closer to a computer's innards in those days; so maybe that explains why he likes to see how the machine handles the numbers inside.

Graham Wideman provides hardware and software engineering support to microcomputer-based research projects, primarily at San Diego State University. (He's also been the editor of *Electronics Today—Canada.*) The research projects include areas as diverse as whale hearing, music-synthesizer algorithms, and the behavior of primates. (We don't think that includes the behavior of one species of primate when they get hold of software that doesn't work as they expected.)

Supplier Notes

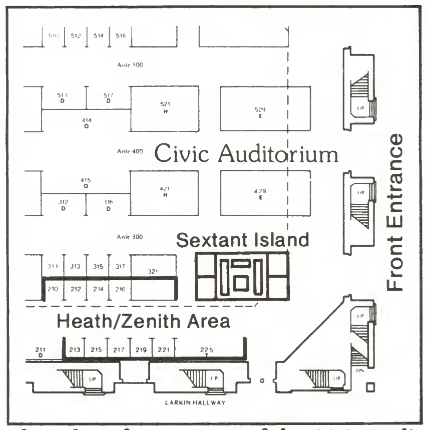
Heath/Zenith Area Biggest Ever at Ninth West Coast Computer Faire

Independent support for Heath/Zenith computers will be present in force at the Ninth West Coast Computer Faire, March 22–25 in San Francisco.

The Heath Company's booth will be surrounded by exhibits by supporting companies, including Kres Engineering, Magnolia Microsystems, Sextant Publishing Company, The Software Toolworks, and Trionyx Electronics.

To expand the size and effectiveness of the Heath/Zenith area, Sextant Publishing Company has rented the 20′ × 40′ island (booth #329E) across the aisle from Heath's booth. In turn, several independent suppliers have rented space within the booth from Sextant, including C.D.R. Systems Inc., Hudson and Associates, Micro Peripherals Inc. (MPI), Technical Micro Systems Inc., and Microservices.

Suppliers exhibiting in the Sextant island chose from several plans for renting exhibit space. As the diagram below shows, some rented a full 10′ × 10′ area at one corner of the island. Others rented table space by the square foot in the large (20′ × 20′) central area.



Floorplan of one corner of the Civic Auditorium, showing the Heath/Zenith area and the Sextant island.

Some companies unable to attend the Faire will send promotional material to be displayed on a brochures table in the middle of the island. The display table will be maintained by the Sextant staff.

The West Coast Computer Faire is one of the oldest and largest microcomputer trade shows—and is considered by many to be the most important. Exhibits will cover two floors of the San Francisco Civic Auditorium and Brooks Hall. Attendance this year is expected to top 60,000.

The concept of bringing together Heath/Zenith suppliers in one exhibit area at the West Coast Computer Faire was first tried in 1982. That year, *Buss: The Independent Newsletter of Heath Co. Computers*, rented a 20′ × 20′ island and shared it with other Heath/Zenith vendors. The Heath Company provided essentially the same service in 1983, reserving blocks of space and making them available to the companies which support Heath/Zenith systems.

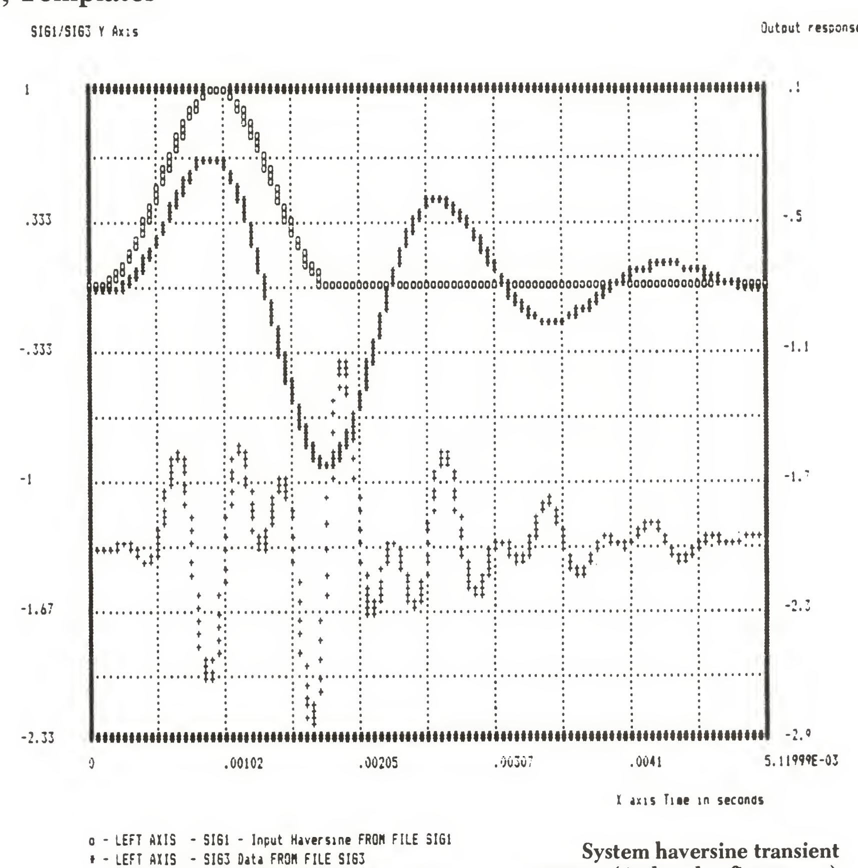
This year, the efforts of the Heath Company and Sextant will combine to create the largest, most impressive display of Heath/Zenith computer strength and support Faire-goers have seen to date.

Plotpro Creates Scientific Graphs, Templates

Plotpro is a set of three Microsoft BASIC programs which make scientific graphs on any 80- to 132-column printer. Plotpro creates linear, semi-logarithmic, and full logarithmic plots and will plot multiple functions on the same graph. Forced scaling and auto-scaling are supported as well as optional grid lines to aid in graph interpretation. Plotpro uses control characters to take advantage of more capable printers such as the Epson to improve the resolution of graphs.

The Protemp model creates templates of the physical appearance of any graph. These templates specify the type of scaling (linear, log, forced, or autoscaled), axis labeling, ranges for each axis, user-specified control characters, and other information. Templates are saved to disk files to develop a library of frequently used templates for future use.

The Proquick module controls plotting and printing of infinite-length graphs limited only by paper length. Plotpro also generates vertical formats suitable for use in reports, viewgraphs, etc. Plotpro does not require special plotters, complicated data interfaces, assembly-language "hooks," or reserved graphics memory. Plotpro data files are easily generated by BASIC, FORTRAN, Pascal, or assembly-language programs. A 20-page manual and sample files are included.



System haversine transient response. (Reduced to fit on page.)

Plotpro is available for Z89s and Z100s running CP/M on 5.25″ double-density, single-sided disks. It is priced at $49.95 for Plotpro and a 20-page manual, with a money-back guarantee. For further information, contact Wilda Daggett, BV Engineering, P.O. Box 3351, Riverside, CA 92519, 714/781–0252.

Support and More

From the Heath/Zenith Compatibility Leaders

H8 PRODUCTS

The Most Extensive Line of Hardware Support for the H8®

- **DG-80/FP8**
  Z80® based CPU including the powerful FP8 monitor — both only $199.00. The acclaimed FP8 monitor package is included with the DG-80 CPU.
- **DG-64D/64K RAM Board**
  Reliable, Low Power, High Capacity Bank-selectable RAM
  Priced from $233.00 (0K) to $299.00 (64K)
- **DG Static 64**
  Fully Static, High Capacity, Bank-selectable RAM. Also can be used as EPROM/PROM board (2716 type EPROMS). Priced from $199.00 (0K) to $499.00 (64K).
- **DG-32D/32K RAM Board**
  Low cost, Dependable RAM for the H8 32K Version Only $179.00.
- **DG-ADP4**
  H17-4 MHz disk adaptor — $19.95

THE SUPER 89

The DG SUPER 89 is a replacement central processor board for the Heath/Zenith 88-89 series of computers. The DG SUPER 89 offers advanced features not available on the standard Heath/Zenith 88-89 such as 4 MHz operation, real-time clock, optional AM9511A arithmetic processor, up to 256K of bank selectable RAM with parity check, and HDOS, CP/M and MP/M compatibility. By incorporating current state-of-the-art technology available for the Z80, the DG SUPER 89 offers the user increased speed and system reliability for years to come. Full compatibility with all Health/Zenith software and hardware products is designed into the DG SUPER 89. Electronic Disk Software included. Priced from $829.00 (128K) to $989.00 (256K).

HEARTBEAT

The DG Heartbeat is a compact computer system designed to be hardware and software compatible with the popular Heath/Zenith Z89/90 computer product line. The Heartbeat offers advanced features not found on the standard Heath/Zenith computer such as 4 MHz operation, real-time clock/calendar, two RS-232 serial ports, five peripheral expansion slots, 128 Kbytes (expandable to 256 Kbytes) parity checked RAM and provisions for an optional AM9511 Arithmetic Processor. Compatible with HDOS, CP/M and MP/M II (Multi-user) operating systems. Electronic Disk Software included. The Heartbeat may be used with most popular video terminals on the market although the Heath/Zenith H/Z19, H/Z29 and ZT-10/11 video terminals are recommended for full Heath/Zenith software compatibility. The Heartbeat cabinet design provides for inclusion of hard and/or floppy disk drives as well as other desired peripheral interfaces and is color-coordinated for use with the Zenith Z29 and ZT-10/11 video terminals. Priced from $1350.00 (Basic Unit).

D·G ELECTRONIC DEVELOPMENTS CO.

Ordering Information: Products listed available from DG Electronic Developments Co . 700 South Armstrong Denison. Tx 75020 Check Money Order. VISA or MasterCard accepted Phone orders call (214) 465-7805 Freight prepaid Allow 3 weeks for personal checks to clear Texas residents add 5% Foreign orders add 30% Prices subject to change without notice

Circle #114 on Reader Service Card

Standard Operating Procedure

Disk and File Management

It is easier to copy a file before it's accidentally erased than afterwards. (Disks, too.)

Dean Gibson

As you may have already found out, a computer tends to magnify the degree of organization that you have in your personal life.

That is, if you are organized in your personal affairs, then the way that you use your computer, and the way that you organize your files on your computer, will tend to be very organized. More importantly, if you are disorganized in your personal affairs, then the way that you use your computer, and the way that you organize your files on your computer, will tend to be very disorganized. And this will frequently lead to disasters: programs and data accidentally destroyed, missing or misplaced programs and data, etc.

Regardless of which type of person you are (organized or disorganized), it pays to spend some time organizing how you operate your computer system.

So let's consider two aspects of organization in the operation of your computer system: first, which files go on which disks; and second, file backup.

In discussing these points, I will assume that your computer system has two disk drives. It is hard to get anything done on a computer with only one disk drive; and most of my comments are applicable only to systems with two or more disk drives. Further, I will also assume that most of you have 5¼″ disk drives. (The larger storage capacity of 8″ and hard disks may reduce the day-to-day pressure for some of the discipline imposed on the users of 5″ drives. However, damage to one 8″ disk will usually incur a much greater loss than a damaged 5″ disk. So the points discussed here are still important for users of the larger disks.)

Also, I will tend to minimize mention of features specific to the various operating systems most of us use. I minimize the operating-system features mainly because the principles that I will be discussing are generally applicable to all computer operating systems. (I use CP/M in my microcomputer consulting work; the rest of the time, I use the Heath Disk Operating System—HDOS.)

Which files on which disks

Most of us have already decided which files go on which disks. The organization I am going to describe is the one that I use, adapted slightly. Because of the size of your disks and the number of program and data files that you may have, you may need to adapt my organization slightly or extensively.

Ask yourself how much time you are willing to spend reconstructing or re-keying lost data.

However, the organization that I use not only assists me in locating a file that I want, but also assists me in backup and recovery, as I will describe later.

First of all, I divide disks into two categories: bootable system disks; and non-bootable "working" disks.

I would suggest keeping two or three bootable system disks. The first bootable disk would contain the operating system (CP/M, HDOS, or the Zenith Disk Operating System—Z-DOS) fully configured for all hardware devices that you have. It would also contain all operating-system utilities. (For CP/M, that would include, for instance, PIP, STAT, MOVCPM, SYSGEN, and CONFIGUR—called SETUP on Magnolia Microsystems CP/M; for HDOS: PIP, FLAGS, SET, INIT, SYSGEN, TEST17; and similarly for Z-DOS.)

This disk would always be used to create a new bootable system disk. If there is enough room, this disk could be combined with the second bootable disk.

The second bootable disk would be the system disk that you would normally boot. It would contain the operating system (CP/M, HDOS, or Z-DOS) fully configured for *all* hardware devices that you have. It would contain *frequently* used utilities (such as PIP or STAT) and those other programs you frequently use (e.g., a compiler/assembler, BASIC, or an editor or word processing program).

The third bootable disk would be booted when you needed the maximum amount of random-access memory (RAM) for a program. This disk might contain the operating system configured to use a minimum amount of RAM; the operating system would be configured for only those hardware devices that you *commonly* use. Like the second disk, it would have only frequently used utilities. And it would contain those programs that manipulate files which need a large amount of memory. (These, too might be the same as the second disk. A compiler, BASIC, and a word processing program may all need to call upon a large amount of memory. You would use the second disk when they handle smaller documents or programs; the third when they work on large ones.)

All of your other disks would be non-bootable, and would be organized by function. For example, disks containing letters that you have typed, BASIC program disks, assembler program disks, game disks, etc.

With this system, then, we would have two types of disks. The contents of the bootable disks would not change much, simplifying backup of them. The original distribution disks serve as the backup to the first bootable disk. It, in turn, serves as backup for the other two. (Although a duplicate of each might still come in handy.)

The data and program disks would be subject to extensive changes and hence need to be backed up accordingly.

This brings me to my next subject: backup.

Backing up files

The purpose of backup is to protect wanted programs and data from being lost or destroyed. As you know, there are two types of errors: human errors and machine errors. Backup is intended to protect against both.

An effective backup system requires a plan. When an error occurs, it may detected immediately, or it may be detected later, perhaps months later, when the data or program is next used.

If all errors were detected immediately, backup plans would be relatively simple. However, what good does it do to back up your disks weekly, saving the last two months worth of backups, only

to find out that you somehow lost some important data three months ago?

The first rule of a backup plan is to use whatever features the operating system provides, in combination with any features provided by your word processor or other software.

The next point to remember is that you do not have to back up everything. Consider a selective backup plan: only back up the files that have changed.

There are several simple ways to implement a selective backup plan. For example, what do you do when you alter a file and save the altered file to disk under the old name? Many editors and word processing programs have the ability to give the previous version of a file an extension of ".BAK". If you have this, not using it is false economy.

You can back up files having the .BAK extension using your operating system's ability to copy wildcards (*). Copy all files with extension .BAK to a backup disk. Then delete all .BAK files off the first disk. While this scheme does not provide a "current" backup copy, it is very fast.

Under HDOS, another alternative is to set the "System" flag on all files. When a file is replaced, the "System" flag disappears, showing which files have changed. You can back up files by periodically copying all the non-"system" files.

A third alternative, for those operating systems which maintain a date (CP/M Plus, HDOS, and Z–DOS), is to use the creation date of the file as a means for determining which files have changed. There are many backup programs commercially available which use the date or one of the "user flag" fields in the directory to keep track of changes. They automatically save changed files onto another disk.

Those with lots of room may consider another alternative: simply adopt a file-name convention that provides for perpetual backups. Each new version might receive a new name: NAMEA, followed by NAMEB, NAMEC, and so on. Such a backup, however, leaves all the files on the same disk.

For a selective backup plan to be effective in locating data that you have selectively backed up, it is probably best to copy changed files to an older backup copy of that entire disk, writing over the previous version of those files.

Backing up whole diskettes

Although a selective backup plan is important, you should also consider the need for backing up whole diskettes. In this regard, ask yourself how much time you are willing to spend reconstructing or re-keying lost data. Compare that amount of time with how long it takes you to back up the data. The rule of thumb that I use is to figure the time that it would take to reconstruct or re-key my data. If it exceeds 20 times the time it takes to back up all my data (that is, my non-bootable disks), I do a complete backup of the entire diskette. I do selective backup more frequently.

How long should you save your backup diskettes? The best plan is to save them forever, or for at least a couple of years. However, that usually is not feasible.

So you may wish to consider the following schedule. Assume that you create a backup diskette for a given disk each week. Save each weekly backup diskette for a month. Once a month, relabel the oldest weekly diskette as a monthly backup diskette; save monthly backup diskettes for three months. Once a quarter, relabel the oldest monthly diskette as a quarterly backup diskette; save quarterly backup diskettes for a year. Once a year, relabel the oldest quarterly diskette as a yearly backup diskette; save yearly backup diskettes for three years, or forever.

Your backup diskettes would then have the following history:

One-week-old backup
Two-week-old backup
Three-week-old backup
One-month-old backup
Two-month-old backup
Three-month-old (one-quarter-old) backup
Two-quarter-old backup
Three-quarter-old backup
Four-quarter-old (one-year-old) backup
Two-year-old backup
. . .

With this plan, a disk that is backed up on a weekly basis only requires nine diskettes to cover a year's worth of history. Some disks, however, may only need to be backed up on a monthly basis. In such a case, only six diskettes are required for a year's history. Once they stop changing, you can stop backing up (e.g., when a letter disk fills up).

Note that this plan serves as an adequate backup for all files on a disk, whether they are changed frequently (in which case, errors would normally be detected quickly), or whether they are changed rarely (in which case an older backup will usually suffice). You can alter this scheme to fit the changeability of your files.

In summary

In summary, remember that good computer operating practices come from organization. Organizing your files on your disks not only can assist you in locating files on disks, but also can assist you in making adequate backup plans.

File organization:

1) Have two or three bootable disks with system utilities and commonly run programs.
2) Group all other programs and data on other diskettes.

A good backup plan:

1) Use operating system or word processor backup features.
2) Consider a selective backup plan.
3) Decide how often you should back up.
4) Decide how long you should save your backup diskettes.

One more item: there is a saying among computer professionals that when you first detect that you have lost or destroyed some data, many times the data has not yet been irretrievably lost. The next thing you do (perhaps as simple as asking for a CATalog) may turn out to be a bigger error than the first. So, before it happens, calm down; write-protect all valuable disks. And proceed slowly.

Scuttlebutt

Pat McNamara, former Zenith Data Systems director of marketing communications, has left ZDS to go to work for **NBI** in Boulder, Colorado. He joins another former ZDS marketing executive, **Robert Reid**. NBI makes dedicated word processors and sells word processing programs for personal computers.

Government employees, including **reservists**, can get a **Z100** for a bargain price. The Air Force and Navy pay **$1,799** for their Z100s, now that the Z100 is the standard desktop computer for the two services. Zenith is selling the '100s through the Government Employees Association **for the same price**. Ordering information will be sent only to GEA members. If you're not a member, send them a request for membership and ordering information. You should include your name, address, agency, and date of birth, as well as a lifetime fee of $10. (GEA, P.O. Box 2405, Arlington, VA 22202, 703/920–5800.)

One source outside GEA told *Sextant* that Zenith has sold "**zillions**" of Z100s through GEA.

Michael Lougee of the University of Michigan's Institute for Social Research dropped by Sextant recently. He reports that there are now at least **350 to 400 Z100s in Ann Arbor**.

Lougee also mentioned that a statistical package called "**ABC**" may be coming out of the University of Michigan this spring. The program, which is intended for people **without** statistical or computing expertise, is running under MS-DOS for the IBM PC and will soon be adapted to run on the Z100.

By the time you read this, you may be able to run PC software like "ABC" on Zenith's newest computer, an IBM PC "clone" expected early this spring.

Zenith is also working on a version of the Z100 with a detachable keyboard and a handle. Apparently it's a bit heavy—so much so that it's been dubbed the "**transpushable.**"

Work is underway to organize a **Western Regional HUG Conference** in Southern California in late 1984. Watch this department and *Buss: The Independent Newsletter of Heath Co. Computers* for further information.

The Software Toolworks recently received some kudos in one of the nation's most influential newspapers, *The Washington Post*. T. R. Reid, one of the personal computer columnists for the weekly *Washington Business* section, chose to inaugurate his column with a guide to spreadsheet "best buys." He singled out The Software Toolworks' Zencalc, at $49.95, as "one of the best software bargains I've ever seen."

We think the availability of **low-cost, high-quality software from the Heath/Zenith community** should be one of Zenith's strongest selling points. For example, we use Steve Robbins' **WatchWord** word processor on the Z-DOS side of our Z100. It has all the features of the $300–$500 programs which are so highly touted for the IBM PC—and WatchWord sells for just $100. (S & K Technology, 4610 Spotted Oak Woods, San Antonio, TX 78249; 512/492–3384.) Another example: we recently received a press release about a $120 program for the PC which transfers files from PC-DOS format to CP/M-86 and vice versa. We transfer files between Z-DOS and CP/M-85 on our Z100 using RDCPM, which is **provided free with Z-DOS**, and another program, J. J. Thompson's ZCOPY, which costs only $21.00 with shipping. (281 Warren Avenue, Kenmore, NY 14217, 716/873–0380 after 5 p.m.) It would be nice if Zenith did more to make potential customers aware of these **H/Z100 software bargains**.

Imitation is the sincerest form of flattery, and one product will let a Kaypro II imitate the H/Z19 graphics set. The Graphics Adapter board enables the Kaypro II to emulate most of the features of the H/Z19 terminal. It's advertised on p. 98 of the November/December 1983 issue of *Pro/Files*, "the magazine for Kaypro users." ($175, Gilderfluke & Company, 120 Midway Drive #013, Anaheim, CA 92085; 714/776–2207.)

Also, software is available which will transfer CP/M files from Kaypro disk format to Zenith disk format, and vice versa. (Disk-Tran: $20 for Zenith to Kaypro, $20 for Kaypro to Zenith; Computer Consultants to Business, 1033 Bishop Walsh Road, Cumberland, MD 21502, 301/759–1260.)

Sextant author **Hugh Kenner** has finished work on a **Z100 book** for Robert J. Brady Co., a division of Prentice-Hall. It's expected early this summer.

Sextant will have a booth in the Heath area at the **Ninth West Coast Computer Faire**, which will be held in **San Francisco, March 22–25**.(See Supplier Notes for more details.) Drop by and say hello—the *Sextant* staff will be there in force.

After the West Coast Computer Faire, we'll be going to New Jersey for the oldest microcomputer conference of them all—the **Trenton Computer Festival**, April 14 and 15.

THE SECRETS OF PERFECT MEMORY: ONE AND ONE HALF EARTH DOLLARS

AT LAST: THE WHOLE TRUTH ABOUT FLOPPIES.

Amazing book reveals all!

How to keep from brainwashing your disk so it never loses it's memory.

How fingerprints can actually damage disks. Unretouched Kirlian photographs of UFO's (Unidentified Floppy Objects)! The incredible importance of making copies: the Department of Redundancy Department—and what goes on when it goes on! Powerful secret methods that scientists claim can actually prevent computer amnesia! All this, and much more . . .

In short, it's an 80-page plain-English, graphically stunning, pocket-sized definitive guide to the care and feeding of flexible disks.

For The Book, ask your nearest computer store that sells Elephant™ disks, and bring along one and one half earth dollars.

For the name of the store, ask us.

Elephant Memory Systems® Marketed exclusively by Dennison Computer Supplies, Inc., 55 Providence Highway, Norwood, MA 02062. Call toll free 1-800-343-8413. In Massachusetts, call collect (617) 769-8150. Telex 951-624.

Circle #130 on Reader Service Card